ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - KI141502

PENGENALAN EMOSI PADA TEKS BERBASIS METODE ENSEMBLE NAIVE BAYES, MAX ENTROPY, DAN KNOWLEDGE-BASED TOOLS

HARI SETIAWAN
NRP 05111440000156

Dosen Pembimbing I
ARYA YUDHI WIJAYA, S.Kom., M.Kom.

Dosen Pembimbing II
RULLY SOELAIMAN, S.Kom., M.Kom.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR - KI141502

PENGENALAN EMOSI PADA TEKS BERBASIS METODE ENSEMBLE NAIVE BAYES, MAX ENTROPY, DAN KNOWLEDGE-BASED TOOLS

HARI SETIAWAN
NRP 05111440000156

Dosen Pembimbing I
ARYA YUDHI WIJAYA, S.Kom., M.Kom.

Dosen Pembimbing II
RULLY SOELAIMAN, S.Kom., M.Kom.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS - KI141502

RECOGNIZING EMOTIONS IN TEXT BASED ON ENSEMBLE METHOD WITH NAIVE BAYES, MAX ENTROPY, AND KNOWLEDGE-BASED TOOLS

HARI SETIAWAN
NRP 05111440000156

Supervisor I
ARYA YUDHI WIJAYA, S.Kom., M.Kom.

Supervisor II
RULLY SOELAIMAN, S.Kom., M.Kom.

Department of INFORMATICS
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

PENGENALAN EMOSI PADA TEKS BERBASIS METODE ENSEMBLE NAIVE BAYES, MAX ENTROPY, DAN KNOWLEDGE-BASED TOOLS

TUGAS AKHIR
Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Dasar Terapan Komputasi
Program Studi S1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :
HARI SETIAWAN
NRP: 05111440000156

Disetujui oleh Dosen Pembimbing Tugas Akhir :

ARYA YUDHI WIJAYA, S.Kom., M.Kom. .....
NIP: 198409042010121002 (Pembimbing 1)

RULLY SOELAIMAN, S.Kom., M.Kom. .....
NIP: 197002131994021001 (Pembimbing 2)

SURABAYA
Juni 2018

(Halaman ini sengaja dikosongkan)

PENGENALAN EMOSI PADA TEKS BERBASIS METODE ENSEMBLE NAIVE BAYES, MAX ENTROPY, DAN KNOWLEDGE-BASED TOOLS

Nama	:	**HARI SETIAWAN**
NRP	:	**05111440000156**
Jurusan	:	**Informatika FTIK**
Pembimbing I	:	**ARYA YUDHI WIJAYA, S.Kom., M.Kom.**
Pembimbing II	:	**RULLY SOELAIMAN, S.Kom., M.Kom.**

Abstrak

Emosi merupakan faktor utama dalam kebiasaan manusia. Cara yang paling sering bagi manusia untuk mengutarakan opini, pemikiran dan berkomunikasi dengan yang lain adalah melalui tulisan. Menganalisis konten web dan pesan teks orang-orang adalah hal yang sangat penting dan menarik. Terdapat banyak hal yang dapat disimpulkan dan dipelajari dari hasil analisis pesan teks manusia, entah itu untuk memperbaiki layanan web yang telah ada, ataupun untuk menambahkan fitur baru yang tepat sasaran untuk pengguna. Dalam proses pengenalan emosi, algoritma Machine Learning seringkali digunakan sebagai pemecahan masalah. Namun, penggunaan satu jenis algoritma saja seringkali tidak cukup baik untuk dapat menyelesaikan permasalahannya.

Dalam tugas akhir ini akan dibuat sebuah rancangan sistem dengan menggunakan metode Ensemble, dimana sistem akan menerima input berupa data teks yang berisi sebuah makna emosi yang kemudian diklasifikasikan dan menghasilkan output berupa jenis emosi yang ada dalam teks itu yaitu positif atau negatif. Metode Ensemble akan menggabungkan dua pendekatan

statistikal yaitu Naive Bayes dan Max Entropy, serta pendekatan leksikal dengan penggunaan Knowledge-based tools. Penggunaan metode Ensemble diharapkan dapat mengatasi kelemahan-kelemahan yang dimiliki oleh tiap jenis algoritma. Ensemble merupakan salah satu metode untuk mendapatkan hasil yang lebih optimal dalam memecahkan suatu permasalahan.

Tugas akhir ini menggunakan dataset publik ISEAR dan AffectiveText untuk tahap evaluasi. Hasil uji coba pada dataset ISEAR mendapatkan nilai rata-rata akurasi sebesar 91,40% dan presisi sebesar 79,80%. Sementara dengan dataset AffectiveText mendapatkan nilai rata-rata akurasi sebesar 70,14% dan presisi sebesar 68,60%. Sedangkan dengan menggabungkan kedua dataset tersebut, didapatkan nilai rata-rata akurasi sebesar 87,34% dan presisi sebesar 76,35%. Dengan hasil uji coba ini metode ini terbukti dapat meningkatkan akurasi klasifikasi dalam 15 kasus.

***Kata-Kunci**: sentiment analysis, pengenalan emosi, text mining, classifiers ensembles, affective computing, machine learning*

RECOGNIZING EMOTIONS IN TEXT BASED ON ENSEMBLE METHOD WITH NAIVE BAYES, MAX ENTROPY, AND KNOWLEDGE-BASED TOOLS

Name	:	**HARI SETIAWAN**
NRP	:	**05111440000156**
Major	:	**Informatics FTIK**
Supervisor I	:	**ARYA YUDHI WIJAYA, S.Kom., M.Kom.**
Supervisor II	:	**RULLY SOELAIMAN, S.Kom., M.Kom.**

Abstract

Emotion is the main factor contributing in human behavior. The most common way for human to express their opinions, thoughts, and communicating each other is by the written text. Analyzing people's web contents and textual message is very important and interesting. There are so many things that you can get from analyzing them, either to fix the existing service or to make a new accurate feature for user. In the process of emotion recognition, most of the Machine Learning algorithm are the problem solver. However, the use of one type of algorithm only is often not good enough to solve the problem well.

In this final thesis there will be created a system using Ensemble method. Ensemble metod will combine two statistical approach, Naive Bayes and Max Entropy, and knowledge-based approach that using Knowledge-based tools. By using this Ensemble method, it is expected to cover some weaknesses whose an algorithm has. Whereas an Ensemble method can use more that one approach or classifier model to solve a problem.

This thesis will be using a public dataset ISEAR and AffectiveText in evaluation phase. From the testing result using

ISEAR dataset, we get an average accuracy number of 91,40% and precision number of 79,80%. While using the AffectiveText dataset, we get an average accuracy number of 70,14% and precision of 68,60%. Finally in the last testing scenario, when we use both of datasets, we get an average accuracy number of 87,34% and precision of 76,35%. With this testing result, we can conclude that this Ensemble method is proved to be increase the classification accuracy in fifteenth case.

***Keywords**: sentiment analysis, emotion recognition, text mining, classifiers ensembles, affective computing, machine learning*

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **PENGENALAN EMOSI PADA TEKS BERBASIS METODE ENSEMBLE NAIVE BAYES, MAX ENTROPY, DAN KNOWLEDGE-BASED TOOLS**. Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari. Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT atas anugerahnya yang tidak terkira kepada penulis dan Nabi Muhammad SAW.
2. Ibu saya tercinta yang selalu mendukung saya.
3. Bapak Arya Yudhi Wijaya, S.Kom., M.Kom. selaku pembimbing I yang telah membantu, membimbing, dan memotivasi penulis mulai dari pengerjaan proposal hingga terselesaikannya Tugas Akhir ini.
4. Bapak Rully Soelaiman, S.Kom., M.Kom. selaku pembimbing II yang juga telah membantu, membimbing, dan memotivasi penulis mulai dari pengerjaan proposal

hingga terselesaikannya Tugas Akhir ini.

5. Muhammad Irza Zakaria Asy-Sya'bani dan Shafira Aisyah Ramadhani yang selalu mengingatkan saya pada masa kecil saya. Semoga kalian selalu berada di bawah lindungan Allah SWT.
6. Segenap keluarga administrator Laboratorium Pemrograman(Raca, Nafiar, Mala, Afiif, Rina, Rizka, Ghisa, Sekbay, Yuuta, Brian, Bonbon, Dita, Nobby, Irsyad, Mike, John) yang telah menemani saya dan memberikan banyak semangat selama pengerjaan Tugas Akhir ini.
7. Seluruh anggota OrangSibuk yang selalu memberikan candaan dan berbagi kebahagiaan bersama sejak awal perkuliahan.
8. Semua teman-teman TC14 yang sudah memberi saya semangat sampai saat ini.
9. Serta semua pihak yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Juni 2018

Hari Setiawan

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1 Latar Belakang

Kognisi dan emosi manusia adalah bawaan sejak lahir dan sangat berarti dalam aspek alami manusia. Penelitian dalam bidang Kecerdasan Buatan mencoba untuk mengeksplorasi dan mendapatkan pemahaman lebih baik dari mekanisme di balik kebiasaan manusia yang bertujuan untuk memberikan sistem dan aplikasi komputer kemampuan untuk mengenali aspek–aspek alami dari manusia, seperti emosi. Emosi merupakan faktor utama kecerdasan manusia yang memberikan karakteristik indikatif perilaku manusia, mewarnai cara komunikasi manusia dan dapat memainkan peran penting dalam interaksi komputer dan manusia.

Peran dari emosi pertama kali diteliti oleh Picard, yang memperkenalkan konsep komputasi afektif, menunjukkan pentingnya emosi dalam interaksi komputer manusia dan menggambarkan arah untuk penelitian interdisipliner dari berbagai bidang, seperti ilmu komputer, ilmu kognitif, dan psikologi. Tujuan komputasi afektif adalah untuk memungkinkan komputer mengenali status emosi dan perilaku manusia dan menjembatani kesenjangan antara emosi manusia dan komputer dengan mengembangkan sistem dan aplikasi yang bisa menganalisa, mengenali, dan beradaptasi sesuai dengan keadaan emosi pengguna.

Emosi manusia dapat diekspresikan melalui berbagai media,

seperti ucapan, ekspresi wajah, gerak tubuh, dan data tekstual. Cara yang paling umum bagi orang-orang untuk berkomunikasi dengan orang lain dan dengan sistem komputer adalah melalui teks, yang merupakan bentuk komunikasi utam baik itu dalam web ataupun sosial media. Selama beberapa tahun terakhir kemunculan web dan maraknya sosial media telah benar–benar mengubah cara komunikasi manusia karena mereka menyediakan sarana baru yang menghubungkan semua orang di seluruh dunia dengan informasi, acara, dan berita secara *real time*. Selain itu, mereka telah benar–benar mengubah peran pengguna; mereka telah mengubah peran pengguna dari pencari informasi pasif sederhana dan konsumen untuk produsen aktif [1]. Setiap hari, sejumlah besar artikel dan pesan teks ditulis di berbagai situs, blog, portal berita, toko online, jaringan sosial dan forum. Jumlah konten tekstual pada web memerlukan metode otomatis untuk menganalisa dan mengekstrak pengetahuan[2].

Menganalisa konten web dan pesan tekstual masyarakat dengan tujuan untuk menentukan status emosi mereka adalah topik yang sangat menarik dan menantang di area *microblogging* . Aliran data tekstual yang besar dan terus menerus di web dapat mencerminkan perasaan, pendapat, dan pemikiran para penulis di berbagai fenomena mulai dari peristiwa politik di seluruh dunia. Hal itu bisa menyampaikan status emosi seseorang dan informasi substansial tentang keyakinan dan sikap mereka. Analisis data tekstual diperlukan untuk lebih dapat lebih dalam memahami status dan perilaku emosional seseorang, dan dalam garis ini dapat memberikan faktor yang sangat indikatif terhadap sikap masyarakat menuju berbagai topik dan juga dapat menggambarkan status emosional dari sebuah komunitas. kota, atau bahkan negara. Dari sudut pandang manusia sendiri, menganalisis pesan teks dari orang yang spesifik dapat memberikan faktor yang sangat indikatif mengenai situasi emosional orang tersebut, perilaku orang tersebut, dan juga

memberikan petunjuk lebih dalam untuk menentukan kepribadiannya. Selanjutnya, sehubungan dengan berita, artikel dan komentar dari orang–orang, dengan topik yang berkaitan, analisis dari komentar–komentar orang pada sebuah topik spesifik dapat memberikan informasi yang sangat bermanfaat tentang sikap publik, perasaan dan perilakunya mengenai berbagai topik dan acara. Pada hal ini, model emosi dapat digunakan untuk mengetahui apa yang orang rasakan mengenai sebuah entitas yang diberikan seperti film, ataupun acara tertentu.

Namun, pengembangan dari sistem dan aplikasi untuk menganalisis bahasa alami secara otomatis dengan tujuan untuk mengetahui isi sentimentalnya adalah proses yang sangat sulit. Beberapa penelitian telah menunjukkan bahwa menganalisis dan mengetahui emosi dari dokumen teks sangatlah kompleks dan merupakan permasalahan pada bidang *NLP (natural language processing)* secara keseluruhan.

Dalam tugas akhir ini, saya menyajikan sebuah sistem *ensemble classifier* untuk sentimen analisis dari data tekstual. Skema *ensemble* tersebut bertujuan untuk secara efektif menggabungkan beberapa macam algoritma klasifikasi dan pembelajaran untuk mengatasi kekurangan dari setiap algoritma. Sistem ini terdiri atas tiga algoritma pembelajaran utama, dua diantaranya adalah algoritma statistikal dan sisanya menggunakan *knowledge based classifier tool*. Ketiga algoritma tersebut digabungkan menggunakan sistem voting. Algoritma statistikal yang digunakan adalah *naive Bayes* dan *maximum entropy* dimana dilatih pada data *ISEAR (Internasional Survei on Emotion Antecedents and Reaction)* [3] dan *Affective Text* dataset. Sedangkan *Knowledge based tool* menganilisis struktur kalimat menggunakan peralatan seperti *Stanford parser* untuk mengetahui kata yang dibutuhkan dan menggunakan *WordNet Affect*, sumber leksikal untuk mengetahui kata yang mewakili emosi–emosi yang dicari. Kemudian dari kata–kata tersebut

dapat diketahui kekuatan emosional setiap kata dan menentukan status emosional kalimat tersebut berdasarkan grafik ketergantungan kalimat itu dalam pendekatan dimana keseluruhan keadaan emosi suatu kalimat didapatkan dari kedekatan bagian–bagian emotional dari kalimat tersebut. Sistem klasifikasi ini melakukan analisis sentimen pada tingkat kalimat, oleh karena itu kalimat yang baru pada awalnya akan dipisah menjadi beberapa kalimat dan kemudian baru dimasukkan ke dalam sistem klasifikasi ini, dimana kita akan mengekstrasi fitur–fiturnya, yang direpresentasikan dalam bentuk *bag-of-words*, dan kemudian dikirim kepada algoritma klasifikasi statistikal. Sistem klasifikasi *ensemble* ini menentukan apakah sebuah teks mengandung makna emosional atau netral, dan dalam kasus jika dia emosional maka akan ditentukan emosi apa yang terdapat didalamnya.

Hasil tugas akhir ini diharapkan dapat memberikan metode yang lebih baik untuk deteksi emosi pada teks sehingga dapat menyelesaikan permasalahan di atas dengan optimal dan diharapkan dapat memberikan kontribusi pada perkembangan ilmu pengetahuan dan teknologi informasi.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut :

1. Bagaimana melakukan deteksi emosi menggunakan metode *ensemble*?
2. Bagaimana melakukan deteksi emosi menggunakan metode *ensemble* untuk mengurangi jumlah emosi yang salah terdeteksi?
3. Bagaimana menghitung akurasi deteksi emosi menggunakan metode *ensemble*?

1.3 Batasan Masalah

Dari permasalahan yang telah diuraikan di atas, terdapat beberapa batasan masalah pada tugas akhir ini, yaitu:

1. Metode untuk deteksi emosi yang digunakan dengan menggunakan metode *ensemble* yang menggabungkan algoritma *naive bayes*, *max entropy*, dan *knowledge based tool*.
2. Percobaan akan dilakukan menggunakan Python.
3. Dataset yang digunakan untuk percobaan adalah dataset yang tersedia untuk umum (publik) yaitu dataset yang diambil dari *International Survey on Emotion Antecedents and Reaction* dan *Affective Text* dataset.
4. Perbandingan akurasi yang dilakukan pada metode *ensemble* hanya dilakukan terhadap algoritma *naive bayes* dan *max entropy*.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini yaitu untuk mengetahui akurasi pada metode *ensemble* yang menggabungkan algoritma *naive bayes*, *max entropy*, dan *knowledge based tool*.

1.5 Manfaat

Tugas akhir ini diharapkan dapat membantu memberikan metode yang lebih baik dari beberapa metode deteksi emosi pada teks yang telah ada.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

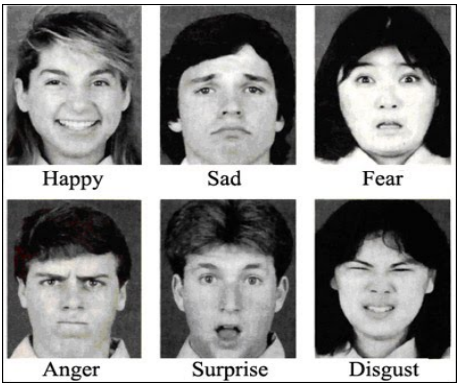
2.1 Model Emosi

Emosi adalah sebuah perasaan kuat yang didapatkan dari sebuah situasi, suasana hati atau hubungan dengan orang lain. Sebuah cara bagaimana emosi dapat direpresentasikan adalah sebuah aspek dasar dari sistem pengenalan emosi. Model yang paling populer untuk merepresentasikan emosi adalah model kategorikal dan dimensional. Model kategorikal mengasumsikan bahwa terdapat beberapa emosi diskrit dan dasar dengan jumlah yang terbatas, dimana setiap emosi tersebut melayani tujuan tertentu. Disisi lain, model dimensional memiliki cara yang berbeda dan merepresentasikan emosi dalam sebuah pendekatan dimensional.

Model kategorikal yang paling populer dan banyak digunakan adalah model emosi Ekman, dimana membagi emosi dasar manusia menjadi: marah, menjijikkan, ketakutan, kebahagiaan, kesedihan, dan terkejut. Emosi–emosi ini dikarakteristikkan sebagai *universal*, sebagaimana mereka diekspresikan dalam cara yang sama tanpa memandang perbedaan budaya dan era. Gambar 2.1 menunjukkan jenis-jenis emosi pada model emosi Ekman.

2.2 *Sentiment Analysis*

Sentiment analysis, juga bisa disebut *opinion mining* adalah bidang studi yang menganalisis opini orang-orang, sentimen, evaluasi, sikap dan emosi terhadap sebuah entitas seperti produk, layanan, organisasi, individu, isu, kejadian, topik, dan atribut mereka. Hal ini memiliki lingkup permasalahan yang luas. Selain itu, juga terdapat beberapa nama dan tugas terkait, contohnya *sentiment analysis, opinion mining, opinion*



Gambar 2.1: Model Emosi Ekman

extraction, sentiment mining, subjectivity analysis, emotion analysis, review mining, dan lain-lain. Namun, mereka semua masih berada dibawah lingkup *sentiment analysis* atau *opinion mining*. Sementara itu di dunia industri, istilah *sentiment analysis* lebih sering digunakan, sedangkan di dunia akademis *sentiment analysis* dan *opinion mining* keduanya sama-sama sering digunakan.

Meskipun linguistik dan *natural language processing*(NLP) memiliki sejarah yang panjang, hanya sedikit penelitian yang telah dilakukan mengenai pendapat dan sentimen orang-orang sebelum tahun 2000. Setelah itu, bidang ini baru menjadi sangat aktif dalam bidang penelitian. Terdapat beberapa alasan untuk ini. Pertama, bidang ini memiliki jangkauan aplikasi yang sangat luas, hampir di semua domain. Kedua, bidang ini menawarkan sangat banyak permasalahan penelitian yang menantang, dimana belum pernah ada yang mempelajarinya sebelumnya. Ketiga, untuk pertama kalinya dalam sejarah manusia, kita sekarang memiliki volume yang sangat besar dari data-data yang berisi opini dalam sosial media dalam Web. Tanpa ini, banyak penelitian yang tidak akan mungkin terjadi. Tidak kaget, bila

lahir dan pesatnya perkembangan dari *sentiment analysis* bertepatan dengan perkembangan sosial media. Faktanya *sentiment analysis* adalah pusat penelitian sosial media untuk saat ini.

2.3 Metode *Ensemble*

Metode ensemble adalah sebuah cara untuk menggunakan lebih dari satu algoritma pembelajaran untuk memperoleh performa prediktif yang lebih baik dari pada satu algoritma apapun. Tujuan dari ensemble adalah untuk mengambil keuntungan dari kelebihan salah satu algoritma dan meminimalkan dampaknya.

Ada banyak alasan untuk mendesain, mengembangkan, dan menggunakan sistem klasifikasi *ensemble* seperti disebutkan oleh *Dietterich*. Dari pandangan statistikal, dengan membangun skema *ensemble* dari *classifier* yang telah dilatih, algoritma tersebut dapat mengambil rata–rata dari vote mereka dan mengurangi risiko kesalahan. Meskipun menggunakan *classifier* yang berbeda dan menghasilkan performa yang baik, ketika hanya satu yang dipilih, hasilnya tidak akan memiliki kemampuan generalisasi yang baik untuk data yang baru. Secara teoretikal, jika model satu *classifier* yang dipilih dapat membuat kesalahan independen, maka dapat dibuktikan bahwa pilihan yang mayoritas adalah yang paling cocok dan dapat mengalahkan performa model *classifier* yang terbaik.

2.4 *ISEAR*

Selama beberapa tahun pada 1990, kumpulan psikolog di seluruh dunia mengumpulkan data dalam proyek *ISEAR((International Survey on Emotion Antecedents and Reaction)*, yang diarahkan oleh Klaus R.Scherer dan Harald Walllbott. Para responden, psikolog dan yang bukan psikolog diminta untuk melaporkan situasi dimana mereka telah

mengalami semua dari 7 emosi utama(*joy, fear, anger, sadness, disgust, shame,* dan *guilt*). Tabel 2.1 menunjukkan beberapa contoh teks pada dataset ini.

Tabel 2.1: Contoh dataset *ISEAR*

Label Emosi	**Contoh Teks**
joy	During the period of falling in love, each time that we met and especially when we had not met for a long time.
fear	When I was involved in a traffic accident.
anger	When I was talking to HIM at a party for the first time in a long while and a friend came and interrupted us and HE left.
sadness	When I lost the person who meant the most to me.
disgust	When I saw all the very drunk kids (13-14 years old) in town on Walpurgis night.
shame	When I did not speak the truth.
guilt	When I caused problems for somebody because he could not keep the appointed time and this led to various consequences.

2.5 *AffectiveText*

AffectiveText adalah sebuah dataset yang berisi kumpulan teks-teks pendek. Kebanyakan dari teks ini adalah *headline* dari berita. Teks-teks ini telah diberi label sesuai berdasarkan emosi yang terdiri atas *anger, disgust, fear, joy, sadness* dan *surprised*. Tabel 2.2 menunjukkan beberapa contoh teks pada dataset ini.

Tabel 2.2: Contoh dataset *ISEAR*

Label Emosi	**Contoh Teks**
joy	Goal delight for Sheva.
fear	Bombers kill shoppers.
anger	UK announces immigration restrictions.
sadness	Mortar assault leaves at least 18 dead.
disgust	Bush Insists Troops Stay in Iraq, Predicts Midterm Victory.

2.6 *Naive Bayes classifier*

Dalam *machine learning, Naive Bayes classifier* adalah keluarga dari sistem klasifikasi probabilistik sederhana berdasarkan penerapan teori *Bayes* dengan asumsi independen yang kuat antar fitur. *Naive Bayes* adalah teknik sederhana untuk membangun sebuah *classifier*, dimana menetapkan label kelas pada tiap data permasalahan, direpresentasikan sebagai vektor berisi nilai fitur, dimana label–label kelasnya diambil dari beberapa kumpulan data terbatas.

Untuk beberapa tipe dari model probabilitas, *naive Bayes classifiers* dapat dilatih dengan sangat efisien pada pengaturan *supervised learning*. Dalam banyak implementasi, estimasi parameter untuk naive Bayes model menggunakan metode *maximum likelihood*; dengan kata lain, kita bisa menggunakan *naive Bayes* tanpa menerima probabilitas Bayesian. Meskipun desain dari sistem klasifikasi ini tampaknya terlalu sederhana, *naive Bayes classifier* telah bekerja cukup baik pada banyak situasi kompleks dunia nyata. Sebuah keuntungan dari *naive Bayes* adalah kita hanya membutuhkan sejumlah kecil data latihan untuk dapat memprediksi dengan baik.

Secara abstrak, *Naive Bayes* adalah sebuah model

probabilitas kondisional: dimana diberikan sebuah vektor $x = x_1, \ldots, x_n$ yang merepresentasikan beberapa *n* fitur (variabel independen), kita tetapkan itu kedalam instansi probabilitas ini

$$p(C_k|x_1, ..., x_n) \tag{2.1}$$

Permasalahan dengan formulasi diatas adalah jika jumlah dari fitur n besar atau jika sebuah fitur dapat memiliki nilai yang besar. Oleh karena itu, formula probabilitas kondisional diatas dapat didekomposisi menjadi

$$p(C_k|x) = \frac{p(C_k)p(x|C_k)}{p(x)} \tag{2.2}$$

2.6.1 *Multinomial Naive Bayes*

Salah satu jenis dari model *Naive Bayes* yang sering digunakan pada permasalahan klasifikasi teks adalah *Multinomial Naive Bayes*. Dengan probabilitas dari sebuah dokumen *d* menjadi kelas *c* dihitung dengan

$$P(c|d)\alpha P(c) \prod_{1 \leq k \leq n_d} P(t_k|c) \tag{2.3}$$

di mana $P(t_k|c)$ adalah probabilitas kondisional dari kata t_k yang muncul pada dokumen dari kelas *c*. Kita gunakan $P(t_k|c)$ sebagai ukuran seberapa banyak bukti t_k menambah probabilitas bahwa *c* adalah kelas yang benar untuk dokumen tersebut. $P(c)$ adalah *prior probability* dari dokumen yang muncul dalam kelas *c*. Jika kata-kata sebuah dokumen tidak memberikan buktiyang jelas untuk satu kelas terhadap kelas yang lainnya, maka kita pilih salah satu kelas yang memiliki *prior probability yang lebih tinggi*. $< t_1, t_2, ..., t_{n_d} >$ adalah token-token pada *d*. Contohnya, $< t_1, t_2, ..., t_{n_d} >$ untuk dokumen ”Beijing and Taipei join the WTO” akan menjadi <Beijing, Taipei, join, WTO>, dengan $n_d = 4$, jika kita menganggap kata ”the” dan ”and” sebagai

stopwords.

2.6.2 *Bernoulli Naive Bayes*

Salah satu jenis dari model *Naive Bayes* yang lain adalah model *Bernoulli Naive Bayes*. Di mana perbedaan utama dari model *Bernoulli* dan Multinomial ada pada strategi estimasi dan aturan klasifikasinya. Model Bernoulli menggunakan $P(t|c)$ sebagai fraksi dari dokumen pada kelas *c* yang berisi kata *t*. Sebaliknya, model Multinomial menggunakan $P(t|c)$ sebagai fraksi dari token posisi pada dokumen kelas *c* yang berisi kata *t*.

Ketika mengklasifikasikan sebuah tes dokumen, model Bernoulli menggunakan nilai kemunculan *binary*, yang berarti tidak menghiraukan jumlah kemunculan kata tersebut pada dokumennya, di mana model Multinomial menghitung nilai kemunculan dari tiap kata-katanya. Hasilnya, model Bernoulli biasanya membuat banyak kesalahan ketika mengklasifikasikan dokumen yang panjang.

2.7 *Maximum Entropy classifier*

Maximum Entropy classifier adalah model berdasarkan fitur yang lebih memilih model yang paling seragam untuk memenuhi batasan-batasan yang diberikan. Data yang telah memiliki label pada tahap *training* digunakan untuk mendapatkan batasan-batasan untuk model dalam memahami label kelas tersebut. Berlawanan dengan *Naive Bayes*, *Maximum Entropy classifier* tidak membuat asumsi independen untuk fitur-fiturnya. Oleh karena itu, hal ini memungkinkan untuk menambahkan fitur pada *Maximum Entropy classifier* seperti *unigrams, bigrams*, dan *N-grams* pada umumnya. *Maximum Entropy classifier* dapat menyelesaikan permasalahan-permasalahan klasifikasi yang sangat sulit dan memiliki performa yang baik dalam berbagai tugas *natural language processing* seperti *sentence segmentation, language modeling,* dan *named entity recognition*.

Maximum Entropy classifier juga cocok digunakan bila kita tidak dapat mengasumsikan status independen kondisional dari fitur yang ada, dimana hal ini adalah suatu hal yang sangat benar dalam permasalahan-permasalahan *text mining* dan *sentiment analysis*, dimana fitur kata-kata yang ada tidak independen.

2.7.1 LIBLINEAR *kernel*

LIBLINEAR adalah sebuah pustaka *open source* untuk klasifikasi linear berskala besar. LIBLINEAR mendukung *logistic regression* dan *linear support vector machine*. Di mana terdapat sebuah set pasangan data dengan labelnya $(x_i, y_i), i = 1, ..., l, x_i \in R^n, y_i n{-}1, +1$, kedua metode ini memecahkan permasalahan optimasi berikut dengan *loss function* yang berbeda-beda $\xi(w; x_i, y_i)$:

di mana C > 0 adalah sebuah parameter pinalti. Untuk SVM dua *loss function* yang paling umum adalah $\max(1 - y_i w^T x_i, 0)$ dan $\max(1{-}y_i w^T x_i, 0)^2$. Yang pertama disebut sebagai L1-SVM, sedangkan yang kedua L2-SVM. Untuk Linear Regression, *loss function* yang digunakan adalah $log(1{+}e^{-y_i w^T x_i})$ yang diturunkan dari model probabilistik.

2.8 *Knowledge-based tools*

Pendekatan dengan menggunakan *knowledge-based tools* atau berbasis pengetahuan, berbeda dengan pendekatan berbasis statistik. Dimana pendekatan ini mencoba untuk menganalisis dan mengekstrak pengetahuan/pengertian dari setiap kalimat dengan tujuan untuk menentukan status emosinya. *Knowledge-based tools* ini digunakan untuk mengetahui informasi mengenai kata-kata emosional yang diketahui menunjukkan emosi. Informasi ini berdasarkan sebuah sumber leksikal *WordNet Affect*.

2.8.1 WordNet Affect

WordNet-Affect adalah sebuah ekstensi dari Domain WordNet, yang memiliki kumpulan *synset*(kumpulan sinonim dari sebuah kata) yang cocok untuk merepresentasikan konsep yang berhubungan dengan kata-kata afektif. Sama seperti metode yang digunakan pada label domain, WordNet Affect menetapkan satu atau lebih label(a-label). Khususnya, konsep afektif merepresentasikan status emosional dari tiap *synset* dengan a-label EMOTION. Selain itu, juga terdapat a-label lainnya untuk konsep ini yang merepresentasikan situasi hati, situasi yang memunculkan emosi, atau respon-respon emosional.

Sumber ini dikembangkan dengan kumpulan a-label tambahan(kategori emosional), yang ditata dengan struktur hierarki. Dalam sumber ini digunakan empat a-label tambahan untuk mewakili emosi yang ada yaitu: POSITIVE, NEGATIVE, AMBIGUOUS, DAN NEUTRAL.

Yang pertama sesuai dengan emosi-emosi positif, seperti yang telah disebutkan sebagai status emosional dengan adanya sinyal *edonic* positif(kebahagiaan). Emosi ini juga memiliki *synset* seperti senang atau antusiasme. Begitu juga dengan a-label NEGATIVE mengidentifikasi emosi-emosi negatif yang ditunjukkan dengan adanya sinyal *edonic* negatif(rasa sakit), contohnya marah atau kesedihan. *Synset* yang menunjukkan status afektif dimana nilainya bergantung pada konteks semantiknya ditandai dengan a-label AMBIGUOUS. Dan yang terakhir *synset* yang merujuk pada kondisi mental yang secara umum memiliki emosi namun tidak memiliki nilai, ditandai dengan a-label NEUTRAL. Contoh-contoh a-label beserta *synset*nya ditunjukkan pada tabel 2.1.

Tabel 2.3: A-label dan contoh *synset* yang sesuai

A-label	**Contoh**
EMOTION	noun anger, verb fear
MOOD	noun animosisy, adjective amiable
TRAIT	noun aggressiveness, adjective competitive
COGNITIVE STATE	noun confusion, adjective dazed
PHYSICAL STATE	noun illness, adjective all in
HEDONIC SIGNAL	noun hurt, noun suffering
EMOTION-ELICITING SITUATION	noun awkwardness, adjective out of danger
EMOTIONAL RESPONSE	noun cold sweat, verb tremble
BEHAVIOUR	noun offense, adjective inhibited
ATTITUDE	noun intolerance, noun defensive
SENSATION	noun coldness, verb feel

2.8.2 *TreeTagger*

TreeTagger adalah sebuah alat untuk menandai tiap teks dengan *part-of-speech* dan informasi kata dasarnya. *TreeTagger* dikembangkan oleh Helmut Schmid dalam TC *project* dari *Institute for Computational Linguistics of the University of Stuttgart*. *TreeTagger* telah berhasil digunakan untuk menandai beberapa bahasa yaitu German, English, French, Italian, Danish, Dutch, Spanish, Bulgarian, Russian, Portuguese, Galician, Greek, Chinese, Swahili, Slovak, Slovenian, Latin, Estonian, Polish, Romanian, Czech, Coptic dan lain-lain.

Part of speech adalah bagian-bagian mendasar dari kalimat bahasa inggris. Terdapat 8 *part of speech* utama pada bahasa inggris, yaitu: *noun, pronoun, verb, adjective, adverb, preposition, conjunction, dan interjection*.

TreeTagger dapat beradaptasi dengan bahasa lainnya jika

memang ada kamus dan *training corpus* yang ditandai dengan manual. Tabel 2.2 menunjukkan contoh-contoh *output* dari penggunaan *TreeTagger* pada kalimat ”The TreeTagger is easy to use.”.

Gambar 2.2: Part Of Speech

Tabel 2.4: Contoh *output TreeTagger*

Target kata	***part-of-speech***	***lemma***
The	DT	the
TreeTagger	NP	TreeTagger
is	VBZ	be
easy	JJ	easy
to	TO	to
use	VB	use
.	SENT	.

2.8.3 *Stanford Parser*

Natural language parser adalah sebuah program yang menyusun struktur dari sebuah kalimat secara *grammatical*, contohnya menentukan apakah kumpulan kata-kata yang ada pada sebuah kalimat berperan sebagai frasa atau sebagai subjek atau objek dari kata kerja.*Probabilistic parser* menggunakan pengetahuan dari bahasa tersebut yang didapat dari kalimat-kalimat yang disusun secara manual untuk mendapatkan analisis yang paling mungkin dari kalimat yang baru. *Statistical parser* ini masih membuat beberapa kesalahan, namun seringkali bekerja dengan baik.

Hasil dari *Stanford parser* ini berupa sebuah *dependency tree* dari sebuah kalimat. *Dependency tree* ini adalah sebuah representasi dari sebuah kalimat yang dipecah sesuai dengan perannya pada kalimat tersebut, selain itu tiap *node* memiliki peran khusus terhadap kata-kata di sekitarnya. Gambar 2.2 menunjukkan contoh *output* dari *Stanford parser* pada kalimat ”When I lost the person who meant the most to me”.

2.9 *Natural Language Toolkit (NLTK)*

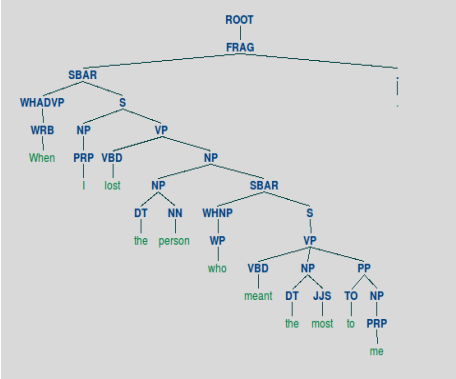
Natural Language Toolkit adalah kumpulan pustaka dan program untuk *symbolic* dan *statistical natural language processing* untuk bahasa inggris yang ditulis dalam bahasa pemrograman Python. NLTK dikembangkan oleh Steven Bird dan Edward Loper di Department of Computer and Information Science di University of Pennysylvania. NLTK juga berisi demonstrasi grafis dan contoh data.

NLTK ditujukan untuk mendukung penelitian dan mengajarkan *natural language processing* atau bidang yang berhubungan, termasuk bidang linguistik empiris, ilmu kognitif, kecerdasan buatan, *information retrieval*, dan *machine learning*. NLTK mendukung klasifikasi, tokenisasi, *stemming, tagging,*

parsing, dan fungsi *semantic reasoning*.

2.10 *Scikit-learn*

Scikit-learn adalah sebuah pustaka *machine learning* gratis yang tersedia untuk bahasa pemrograman Python. *Scikit-learn* memiliki beberapa fungsi yang mendukung tugas-tugas *machine-learning*, contohnya klasifikasi, regresi, *clustering*, beserta algoritma-algoritma lainnya. *Scikit-learn* digunakan bersama dengan pustaka ilmiah Python lainnya, yaitu Numpy dan Scipy.



Gambar 2.3: Output Dependency Tree

(Halaman ini sengaja dikosongkan)

BAB III
ANALISIS DAN PERANCANGAN SISTEM

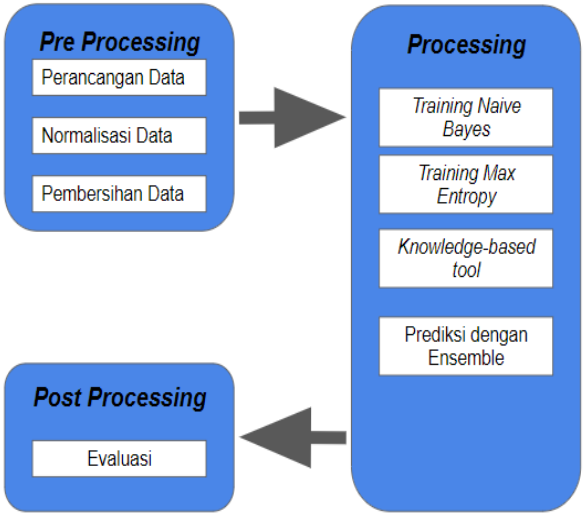
Bab ini akan menjelaskan tentang analisis dan perancangan sistem untuk mencapai tujuan dari tugas akhir. Perancangan ini meliputi perancangan data dan perancangan proses. Bab ini juga akan menjelaskan tentang analisis implementasi metode secara umum pada sistem.

3.1 Analisis Metode Secara Umum

Pada tugas akhir ini akan dibangun suatu sistem untuk melakukan klasifikasi terhadap jenis emosi pada teks menggunakan pustaka Python. Proses-proses yang dilakukan dalam pengimplementasian sistem ini meliputi tahap *Preprocessing*, tahap *Processing* dan Post Processing.

Tahap *Preprocessing* berisi beberapa tahap yaitu Perancangan Data, Normalisasi Data, dan Pembersihan Data. Tahap *Preprocessing* ini dilakukan pada seluruh data sebelum melakukan tahap *Processing* Tahap *Preprocessing* ini adalah salah satu tahap yang paling menentukan kualitas data akan digunakan pada pembuatan model klasifikasi. Tahap yang paling penting pada *Preprocessing* adalah tahap Normalisasi data, dimana pada tahap ini data teks yang masih berisi berbagai tanda baca dan huruf aksen akan dibersihkan menjadi data yang siap untuk digunakan sebagai acuan pembangunan model. Oleh karena itu, jika tahap Normalisasi data ini salah atau kurang tepat, maka model yang dihasilkan juga akan berkurang kemampuannya.

Tahap *Processing* ini berisi proses pembangunan model-model klasifikasi yang akan digunakan pada metode *ensemble*. Dimana ada dua model klasifikasi yang perlu dilatih, yaitu Naive Bayes dan MaxEntropy(Logistic Regression). Selain



Gambar 3.1: Diagram Alir Implementasi Metode Secara Umum

kedua pendekatan statistik tersebut, karena juga menggunakan *Knowledge-based tools* maka juga perlu menginisialisasi kelas-kelas yang dibutuhkan pada *Knowledge-based tool*. Kelas-kelas tersebut adalah *TreeTagger* dan *WNAffect*. Selain itu, yang terakhir adalah proses pemungutan suara dari hasil-hasil prediksi model klasifikasi yang telah dilatih sebelumnya.

Tahap *Post Processing* adalah tahap uji coba menggunakan skenario percobaan yang telah didefinisikan. Setelah mendapatkan hasil-hasil prediksi, gunakan nilai akurasi dan presisi sebagai tolok ukur perbandingan antar hasil skenario. Tahap-tahap utama dari keseluruhan proses ditunjukkan oleh Gambar 3.1.

3.2 Perancangan Data

Subbab ini akan membahas perancangan data yang merupakan bagian penting karena akan menjadi acuan untuk

membuat skenario uji coba dari tugas akhir. *Dataset* yang digunakan adalah data *ISEAR (International Survey on Emotion Antecedents and Reaction)* dan *AffectiveText* yang sudah diberi label emosi sebanyak 8, yaitu *joy, anger, fear, sadness, shame, disgust, guilt,* dan *surprise*. Data memiliki dua fitur utama yang akan digunakan, yaitu jenis emosi dan teks yang berisi emosi tersebut. Untuk skenario uji coba dari tugas akhir ini, klasifikasi dilakukan untuk menentukan apakah teks yang ada memiliki kecenderungan emosi positif atau negatif. Dimana emosi positif terdiri dari label emosi *joy*, sedangkan emosi negatif terdiri dari label emosi *anger, fear, sadness, disgust,* dan *shame*.[4] Gambar 3.2 menunjukkan pembagian label emosi menjadi positif dan negatif.

Gambar 3.2: Pemetaan Emosi Negatif dan Positif

3.2.1 Data *Training*

Data *training* yang digunakan adalah sebanyak 95% dari seluruh data. Dari data *training* yang diperoleh akan digunakan untuk dijadikan model pada metode *Naive Bayes* dan *Max Entropy*. Data yang digunakan pada tahap *training* untuk metode *Naive Bayes* dan *Max Entropy* adalah sebanyak 7423 baris, dimana terdapat sekitar 1034 baris untuk masing-masing kelas.

3.2.2 Data *Testing*

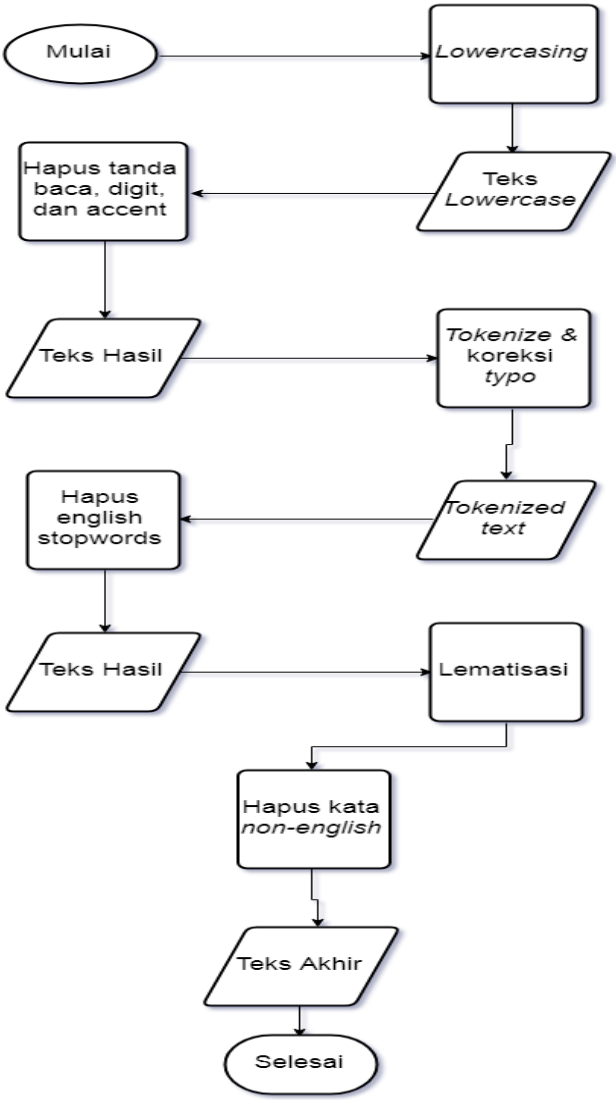
Data *testing* yang digunakan adalah sebanyak 10% dari seluruh data. Data *Testing* inilah yang akan diprediksi menggunakan model *Naive Bayes* dan *Max Entropy* yang telah dilatih dengan Data *training* sebelumnya, dan digabungkan dengan hasil prediksi yang dilakukan dengan *Knowledge-based tool*. Hasil prediksi kelas yang akan diambil ditentukan dengan metode voting pada hasil prediksi tiga metode ini.

3.3 Normalisasi Data

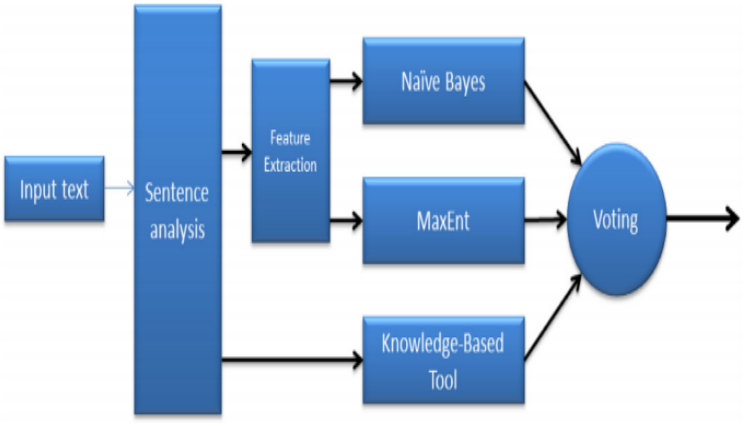
Subbab ini membahas mengenai langkah-langkah yang perlu dilakukan sebelum mulai menggunakan data di dalam algoritma. Langkah-langkah yang dilakukan untuk tiap baris data teks yaitu sebagai berikut :

1. Mengubah semua huruf menjadi kecil
2. Menghapus tanda baca dan aksen
3. Menghapus karakter angka
4. Mengubah tiap kata ke bentuk dasar
5. Menghapus semua kasta *stopwords*
6. Menghapus kata yang memiliki panjang kurang dari 3(kecuali kata ’go’ dan ’ok’)

Gambar 3.3 menunjukkan diagram alir dari seluruh proses Normalisasi Data diatas.



Gambar 3.3: Diagram alir proses Normalisasi Data



Gambar 3.4: Skema Metode *Ensemble*

3.4 Perancangan Proses

Subbab ini membahas mengenai perancangan proses yang dilakukan untuk masing-masing metode yang digabungkan dalam metode *ensemble*. Hasil prediksi dari masing-masing metode ini nantinya akan digabungkan dalam metode *ensemble* untuk mendapatkan hasil yang lebih optimal daripada hanya menggunakan salah satu metode saja. Gambar 3.3 menunjukkan skema *ensemble* yang digunakan.

3.4.1 Metode *Naive Bayes*

Pada metode Naive Bayes, yang digunakan sebagai fitur adalah teks yang berisi emosi. Teks ini dapat memiliki satu kalimat atau lebih. Sebelum dapat digunakan pada algoritma *Naive Bayes*, setiap teks ini diubah terlebih dahulu ke dalam bentuk *vector* yang berisi nilai kemunculan tiap kata, atau yang lebih sering disebut *bag-of-words*. Terdapat dua macam variasi metode *Naive Bayes* yang sering digunakan pada

permasalahan klasifikasi teks, yaitu *Multinomial Naive Bayes* dan *Bernoulli Naive Bayes*. *Multinomial Naive Bayes* mengimplementasikan algoritma *Naive Bayes* untuk data yang terdistribusi secara *Multinomial*, dan metode ini menganggap data yang digunakan direpresentasikan sebagai *vector* yang berisi nilai jumlah kemunculan sebuah kata pada sebuah data. Sedangkan *Bernoulli Naive Bayes* mengimplementasikan algoritma *Naive Bayes* untuk data yang terdistribusi secara distribusi *Multivariate Bernoulli*, dimana terdapat lebih dari satu fitur, namun setiap fiturnya dianggap sebagai variabel bernilai *boolean*. Oleh karena itu, algoritma ini memerlukan fitur yang direpresentasikan sebagai *vector* fitur yang bernilai *boolean*. Namun, algoritma ini akan secara otomatis mengubah fitur-fitur pada datanya menjadi *boolean* jika mereka memiliki nilai yang lain.

3.4.2 Metode *Max Entropy*

Pada metode *Max Entropy* atau yang juga sering disebut *Logistic Regression*, yang digunakan sebagai fitur adalah teks yang berisi emosi sama dengan *Naive Bayes*. Fitur yang digunakan disini juga telah berbentuk *vector* yang berisi *bag-of-words*, dimana setiap fitur berisi nilai frekuensi kemunculan suatu kata pada satu kalimat. Terdapat dua macam variasi algoritma optimisasi untuk metode *Max Entropy* yang digunakan, yaitu *liblinear(coordinate descent)* dan *newton-cg*. Kedua algoritma optimisasi ini meminimalkan nilai kesalahan yang didapatkan dari *loss function*. Adapun persamaan dari *loss function* yang digunakan adalah dibawah ini.

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

Dalam permasalahan ini, model klasifikasi ini meminimalkan nilai kesalahan dengan asumsi *One vs Rest*, dimana model ini

memisahkan tiap satu kelas dengan semua sisa kelas lainnya untuk masing-masing kelas yang ada. Dengan cara ini, algoritma ini menjadikan permasalahan *multiclass* menjadi biner(dua kelas).

3.4.3 *Knowledge-based tool*

Selain menggunakan pendekatan statistikal, yaitu *Naive Bayes* dan *Max Entropy*, juga digunakan pendekatan linguistik dengan *Knowledge-based tool*. Pendekatan ini menggunakan beberapa *library* atau peralatan untuk mengenali bagian-bagian dari suatu teks. Peralatan-peralatan tersebut, yaitu *Stanford Parser*, *Stanford Tree Tagger*, dan sumber leksikal *WordNet Affect*.

Stanford Parser digunakan untuk mendapatkan struktur *dependency tree* dari tiap kalimat. Dengan menggunakan *dependency tree* kita dapat mengetahui bagian-bagian atau *Part of speech* apa saja yang perlu dianalisis untuk mengetahui status emosional dari kalimat tersebut. Dengan ini, kita dapat mempersingkat waktu dengan tidak menganalisis kata-kata yang kurang tepat. Kemudian *Stanford Tree Tagger* digunakan untuk mengambil bentuk baku dan peran *grammatical* dari tiap kata pada sebuah kalimat. Kemudian yang terakhir *WordNet Affect* adalah sebuah sumber leksikal yang berisi daftar kata-kata bahasa inggris yang mengandung nilai emosional didalamnya. Sumber leksikal ini yang paling besar menentukan apakah emosi yang dimiliki oleh sebuah kalimat tersebut. Tabel 3.1 menunjukkan *Part of speech* dari kata-kata yang perlu dianalisis dengan *WordNet Affect*.

Tabel 3.1: *Part of speech* dari kata yang penting

Part of speech	**Keterangan**	**Contoh**
JJ	*Adjective*, kata yang berfungsi memberi keterangan pada kata benda	*happy, dead, sad*
RB	*Adverb*, kata yang berfungsi memberi keterangan pada kata kerja atau kata sifat	*happily, lonely*
VB	*Verb*, bentuk dasar dari kata kerja	*go, angry, live*
VBD	*Verb, past tense* kata kerja lampau, juga termasuk bentuk kondisional *to be*	*were, died*
VBG	*Gerund* atau *present participle*	*working, singing*
VBN	*past participle*	*swollen, broken*
VBP dan VBZ	*present tense*	*work, smoke*

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi sistem pengenalan emosi pada teks secara rinci. Pembahasan dilakukan secara rinci untuk beberapa tahap yang telah disebutkan pada Bab 3. Gambar 4.1 menunjukkan diagram alir implementasi sistem.

4.1 Lingkungan Implementasi

Lingkungan implementasi sistem yang digunakan untuk mengembangkan tugas akhir ini memiliki spesifikasi perangkat keras dan perangkat lunak yang ditunjukkan oleh Tabel 4.1.

Tabel 4.1: Spesifikasi Perangkat

Perangkat	Spesifikasi
Perangkat Keras	• Prosesor: Intel® Core™ i5-4200U CPU @ 1.60GHz (4 CPUs), 2.3GHz • Memori: 8192MB
Perangkat Lunak	• Sistem Operasi Linux Ubuntu Desktop 16.04 LTS • Perangkat Pengembang Python 2.7, MySQL, NLTK, Scikit-Learn, Numpy, • Perangkat Pembantu Sublime Text 3, Microsoft Excel 2016, Microsoft Power Point 2016, HeidiSql

4.2 Implementasi *Preprocessing*

Implementasi *Preprocessing* dilakukan berdasarkan perancangan *Preprocessing* yang dijelaskan pada bab analisis dan perancangan. Untuk implementasi pada tahap *Preprocess* ini dilakukan hanya dalam satu *file*, dimana dibagi dalam beberapa fungsi.

4.2.1 Pengambilan Data

Bagian ini membahas implementasi tahap Pengambilan Data *Affective Text* yang dilakukan pertama kali. Tahap ini diperlukan untuk mengolah data asli yang ada agar dapat digunakan pada metode *Training*. Tahap ini dilakukan dalam fungsi *prepareAffectiveText* dan *processSentence*. Fungsi *processSentence* diperlukan untuk dapat membaca setiap baris data yang ada dan mengambil label emosi yang tepat untuk tiap baris data, kemudian fungsi ini menyimpan tiap baris data ke dalam database. Sedangkan fungsi *prepareAffectiveText* hanya bertugas membuka *file* data yang masih mentah dan menggunakannya pada fungsi *processSentence*.

Saat dijalankan, fungsi *prepareAffectiveText* menyiapkan koneksi pada database *MySQL* yang digunakan terlebih dahulu. Kemudian, menghapus semua data yang ada pada tabel yang sudah ada. Kemudian, fungsi ini membaca tiap *file* yang berisi data nilai emosi dan *file xml* yang berisi data teks untuk dibaca tiap barisnya sebagai *array*. Karena, *file* yang berisi data tiap kalimat berbentuk *xml* maka isi dari *file* ini perlu diproses menggunakan fungsi *ElementTree.parse* terlebih dahulu. Setelah mendapatkan isi dari *file xml* yang berbentuk *array*, gunakan *array* yang berisi kalimat ini, data nilai emosi, dan objek *cursor MySQL* sebagai parameter untuk fungsi *processSentence*. Setelah fungsi *processSentence* selesai, *commit* koneksi yang telah dibuat. Kemudian tutup *cursor* dan koneksi *MySQL* yang telah dibuat.

Fungsi *processSentence* menerima tiga parameter untuk dijalankan, yaitu data nilai emosi, data kalimat yang memiliki emosi tersebut, dan *cursor MySQL*. Fungsi ini melakukan perulangan pada tiap data nilai emosi, kemudian mengubah nilai-nilai tersebut menjadi *array*. Setelah berbentuk *array*, temukan nilai maksimal untuk mengetahui emosi mana yang

paling dominan untuk mewakili kalimat yang ada. Setelah itu, ambil kalimat tersebut dari *file xml*. Setelah mengetahui emosi dan kalimat yang berpasangan, masukkan emosi dan kalimat tersebut ke dalam database *MySQL* agar dapat digunakan pada algoritma *ensemble*. Kode Sumber 4.1 dan Kode Sumber 4.2 menunjukkan pengimplementasian dari tahap Pengambilan Data.

```
import xml.etree.ElementTree as ET
from mysql.connector import MySQLConnection,
Error
from python_mysql_dbconfig import
read_db_config

def prepareAffectiveText():
try:
prefix = 'database/Affective/'
dbconfig = read_db_config()
conn = MySQLConnection(**dbconfig)
cursor = conn.cursor()
cursor.execute("TRUNCATE affectivetext")
with open(prefix+'affective1.emotions.gold') as file:
emo_xml = file.readlines()
trial_count = len(emo_xml)
with open(prefix+'affective2.emotions.gold') as file:
emo_test_xml = file.readlines()
parser = ET.XMLParser(encoding="utf-8")
elem_trial = ET.parse(prefix+'affective1.xml')
elem_test = ET.parse(prefix+'affective2.xml')
elem = elem_trial
emo_xml = emo_xml[1:]
processSentence(emo_xml, elem_trial, cursor)
processSentence(emo_test_xml, elem_test, cursor)
except Error as e:
print(e)

finally:
conn.commit()
cursor.close()
conn.close()
def processSentence(emo_xml, elem, cursor):
  for item in emo_xml:
    emotion_scores = str(item).split(' ')
    scores = []
```

Kode Sumber 4.1: Implementasi Tahap Pengambilan Data Bag. 1

```
scores.append([int(emotion_scores[1]), 'anger'])
scores.append([int(emotion_scores[2]), 'disgust'])
scores.append([int(emotion_scores[3]), 'fear'])
scores.append([int(emotion_scores[4]), 'joy'])
scores.append([int(emotion_scores[5]), 'sadness'])
max_emotion = 0
current_class = ''

for emotion in scores:
if emotion[0] > max_emotion:
max_emotion = emotion[0]
current_class = emotion[1]
if current_class != '':
current_sentence = elem.find('.//instance[@id="' \
+emotion_scores[0]+'"]').text
cursor.execute("INSERT INTO affectivetext(class, \
sentence) VALUES(%s, %s) ", (current_class, \
current_sentence))
return
```

Kode Sumber 4.2: Implementasi Tahap Pengambilan Data Bag. 2

4.2.2 Normalisasi Data

Bagian ini membahas implementasi tahap normalisasi data. Sebelum dapat menggunakan data untuk melatih model *classifier*, data yang digunakan harus diolah dan dibersihkan dari berbagai data yang tidak berguna. Tahap ini membutuhkan beberapa fungsi yaitu *begin*, *preprocess*, dan *getStopwords*.

Sebelum melakukan langkah normalisasi apapun, jalankan fungsi *Begin* untuk mengambil semua data yang telah disimpan ke dalam database *MySQL*. Kemudian, fungsi ini memanggil fungsi *preprocess* dan menggunakan data yang telah diambil serta *cursor MySQL* sebagai parameternya. Setelah itu, fungsi ini memanggil fungsi *registerBow* untuk membangun sebuah tabel *dictionary* yang akan digunakan sebagai acuan untuk membuat fitur *bag-of-words*. Setelah kedua fungsi tersebut selesai, *commit* dan tutup koneksi dan *cursor MySQL* yang telah digunakan.

Setelah mendapatkan semua data, fungsi *registerBow* dijalankan untuk menormalisasikan semua data. Sebelum memeriksa tiap data, muat daftar kata-kata *stopwords* yang telah didefinisikan dalam fungsi *getStopwords*. Kemudian jalankan iterasi untuk memeriksa tiap teks dari data yang telah diambil.

Dalam tiap iterasi, lakukan tiap langkah normalisasi yang telah dijelaskan pada bab analisis dan perancangan. Setelah semua langkah selesai, ubah kelas emosi yang ada menjadi nomor yang telah didefinisikan, yaitu untuk kelas ”joy” menjadi 1, selain itu ubah menjadi 0. Kemudian, simpan tiap data yang telah dinormalisasi kedalam database *MySQL*. Kode Sumber 4.3, Kode Sumber 4.4, dan Kode Sumber 4.5 menunjukkan pengimplementasian dari tahap Normalisasi Data.

```
from mysql.connector import MySQLConnection, Error
from python_mysql_dbconfig import read_db_config
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.tokenize import word_tokenize
import nltk
import re
import sys
from unidecode import unidecode
import string
from autocorrect import spell
from progress.bar import FillingCirclesBar as fcb
from progress.bar import FillingSquaresBar as fsb
from nltk.tag import StanfordNERTagger as nerTagger
import enchant

def begin():
try:
dbconfig = read_db_config()
conn = MySQLConnection(**dbconfig)
cursor = conn.cursor()
cursor.execute("TRUNCATE preprocessed_data")
cursor.execute("SELECT * FROM cleaned_data_original")
isear_row = cursor.fetchall()

all_data = isear_row
preprocess(all_data, cursor)
registerBow(cursor)
  except Error as e:
print(e)
finally:
conn.commit()
cursor.close()
conn.close()
```

Kode Sumber 4.3: Implementasi Tahap Normalisasi Data Bag. 1

```
def preprocess(row, cursor):
stop_words = set(stopwords.words('english'))
new_stopwords = getStopwords()
eng_words = enchant.Dict("en_US")
pb = fcb("Preprocessing words ", max=len(row))
for item in row:
#Removing punctuation and special char
newstring = item[2].lower()
newstring = re.sub('[^A-Za-z0-9 ]+', '', newstring)
#Removing digit
newstring = re.sub('\d+', '', newstring)
newstring = unidecode(newstring)
word_tokens = word_tokenize(newstring)
filtered_sentence = []
last_string = ""
exception_word = ['go', 'ok']
lmtzr = WordNetLemmatizer()
for w in word_tokens:
w = w.lower()
check = False
w = spell(w)
# Lemmatize each words
temp = lmtzr.lemmatize(w, 'a')
if(temp != w):
check = True
if(not check):
temp = lmtzr.lemmatize(w, 'v')
if(temp != w):
check = True
if(not check):
temp = lmtzr.lemmatize(w)
w = temp
if w not in stop_words and w not in new_stopwords \
and eng_words.check(w):
if len(w) > 2:
last_string += " "+ w
filtered_sentence.append(w)
elif len(w) == 2 and w in exception_word:
last_string += " "+ w
filtered_sentence.append(w)
if item[1] == "joy":
new_class = 1
else:
new_class = 0
```

Kode Sumber 4.4: Implementasi Tahap Normalisasi Data Bag. 2

```
if len(filtered_sentence) > 1:
cursor.execute("INSERT INTO preprocessed_data (class,\
sentence) VALUES(%s, %s) ", (new_class, last_string))
pb.next()
pb.finish()
return

def getStopwords():
import glob
file_names = glob.glob("*.txt")
my_stopwords = []
for name in file_names:
with open(name, 'r') as file:
temp = file.readlines()
my_stopwords += temp
return my_stopwords
```

Kode Sumber 4.5: Implementasi Tahap Normalisasi Data Bag. 3

4.3 Implementasi Proses

Implementasi proses dilakukan berdasarkan perancangan proses yang dijelaskan pada bab analisis dan perancangan. Tahap utama dari pengimplementasian proses dibagi menjadi dua, yaitu pembangunan fitur *bag-of-words* dan proses *training* dan *testing*.

4.3.1 Pembangunan fitur *bag-of-words*

Untuk memungkinkan proses pelatihan model *classifier* pada data teks, ubah teks dari tiap baris data yang digunakan menjadi bentuk yang dapat dipahami oleh algoritma *classifier*. Karena model *classifier* tidak dapat membaca teks yang ada, kita ubah tiap kata yang telah disimpan pada tabel *dictionary* sebagai representasi fitur dari tiap baris data. Fitur ini menyimpan nilai kemunculan sebuah kata yang ada pada tabel *dictionary* pada tiap baris teks, atau yang lebih sering disebut *bag-of-words* Tahap implementasi penyimpanan kata ini ada pada fungsi registerBow yang ditunjukkan pada Kode Sumber 4.6.

Untuk membuat fitur *bag-of-words*, muat semua data teks yang telah dinormalisasi ke dalam sebuah *array*. Kemudian untuk tiap baris data, cari tiap kata yang ada pada teks tersebut pada tabel *dictionary*. Kemudian untuk tiap kemunculan kata,

tambahkan nilai kemunculannya pada sebuah *array bag-of-words* sesuai dengan *id* nya pada database sebagai indeksnya dalam *array* tersebut. Kode Sumber 4.7 dan Kode Sumber 4.8 menunjukkan pengimplementasian tahap Pembangunan fitur *bag-of-words* pada fungsi *processWords*.

```
def registerBow(cursor):
cursor.execute("TRUNCATE dictionary")
cursor.execute("ALTER TABLE dictionary AUTO_INCREMENT=1")
cursor.execute("SELECT sentence FROM preprocessed_data")
corpus = cursor.fetchall()
word_list = []
pb = fsb("Registering Bow ", max=len(corpus))
for item in corpus:
words = word_tokenize(item[0])
for w in words:
if w not in word_list:
word_list.append(w)
cursor.execute("INSERT INTO dictionary (word) \
VALUE(%(myword)s)", {'myword': w })
pb.next()
pb.finish()
return
```

Kode Sumber 4.6: Implementasi Tahap Penyimpanan Kata

```
from mysql.connector import MySQLConnection, Error
from nltk.tokenize import word_tokenize
from python_mysql_dbconfig import read_db_config
from sklearn import linear_model
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from progress.bar import IncrementalBar
import numpy as np
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
import knowledge_based as kb
from progress.bar import FillingCirclesBar as fcb
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_recall_fscore_support \
as prf
```

Kode Sumber 4.7: Implementasi Tahap Pembangunan fitur *bag-of-words* Bag.

```
from treetaggerwrapper import TreeTagger
from nltk.parse.stanford import StanfordParser
from nltk.tag import StanfordNERTagger as nerTagger
from nltk.tokenize import word_tokenize
from wnaffect import WNAffect
from emotion import Emotion

prefix = 'knowledge_based/'
def processWords():
try:
dbconfig = read_db_config()
conn = MySQLConnection(**dbconfig)
cursor = conn.cursor(buffered=True)
cursor.execute("SELECT * FROM dictionary")
words = cursor.fetchall()
cursor.execute("SELECT * FROM preprocessed_data")
data_train = cursor.fetchall()
bar = IncrementalBar('Processing Words', \
max=len(data_train))
train_bow = np.zeros((len(data_train), \
len(words)+2), dtype=int)
index = 0
# FIRST COLUMN FOR CLASS, SECOND COLUMN FOR ID
for item in data_train:
sentence = word_tokenize(item[2])
for w in sentence:
cursor.execute("SELECT id FROM dictionary WHERE \
word=%(w)s", {"w": w})
result = cursor.fetchone()
train_bow[index][result[0]+2] += 1
train_bow[index][0] = item[1]
train_bow[index][1] = item[0]
bar.next()
index += 1
bar.finish()
except Exception as e:
print(e)
finally:
conn.commit()
cursor.close()
conn.close()
return train_bow, len(words)
```

Kode Sumber 4.8: Implementasi Tahap Pembangunan fitur *bag-of-words* Bag. 2

4.3.2 *Knowledge-based tools*

Untuk memprediksi emosi yang dimiliki oleh sebuah teks dengan menggunakan *knowledge-based tools*, analisis beberapa hal dari kalimat dari tiap teks. Untuk tahap prediksi diperlukan fungsi *predict*, *lookUp*, *mapEmotions*, dan *findMax*.

Fungsi *predict* berguna untuk menyediakan *interface* bagi metode *ensemble* untuk memprediksi tiap teks. Pada tiap teks yang diberikan, gunakan kelas *TreeTagger* untuk mengetahui *Part of speech* dari tiap kata pada sebuah kalimat. Setelah mengetahui setiap *Part of speech* dari tiap kata, kita hanya perlu menggunakan beberapa kata yang memiliki nilai *Part of speech* yang telah didefinisikan yaitu ”RB, VB, VBD, VBG, VBN, VBP, VBZ, JJ”. Kemudian kita cari status emosional dari kata-kata yang memiliki nilai *Part of speech* tersebut dengan menggunakan fungsi *get_emotion*. Fungsi *get_emotion* ini terdapat pada kelas WNAffect yang telah memuat sumber leksikal pada domain emosi. Setelah mengetahui emosi tiap kata, cari emosi yang paling banyak untuk dijadikan label emosi prediksi. Kode Sumber 4.9 dan 4.10 menunjukkan pengimplementasian fungsi *predict* pada *Knowledge-based tools*.

```
from mysql.connector import MySQLConnection, Error
from python_mysql_dbconfig import read_db_config
from treetaggerwrapper import TreeTagger
from nltk.tokenize import word_tokenize
import os
from collections import defaultdict
from nltk.corpus import wordnet as wn
import treetaggerwrapper
from progress.bar import FillingCirclesBar as fcb
import numpy as np

def predict(target_id, tt, wna, cursor):
cursor.execute("SELECT * FROM cleaned_data_original \
where id = %(target)s", {'target': str(target_id)})
query_res = cursor.fetchone()
if query_res == None:
return 0
sentence = query_res[2]
tags = tt.tag_text(sentence)
```

Kode Sumber 4.9: Implementasi fungsi *predict* pada *Knowledge-based tools* Bag. 1

```
tags = treetaggerwrapper.make_tags(tags)
emotions = ""
check = 0
check_emo = 0

emotion_map = mapEmotions()
emotion_score = np.zeros([2, 7], dtype=int)
relevant_tag = ['RB', 'VB', 'VBD', 'VBG', 'VBN', 'VBP', \
'VBZ', 'JJ']
for i in range(0, 3):
emotion_score[0][i] = i

for word in tags:
if len(word) >= 2 and word[1] in relevant_tag:
emo = wna.get_emotion(word[0], word[1])
if emo != None:
if check_emo == 0:
check_emo = 1
result = lookUp(str(emo), emotion_map)
if result != -1:
emotion_score[1][result] += 1
if check_emo != 0:
result_index = np.unravel_index(np\
.argmax(emotion_score[1], axis=None), \
emotion_score[1].shape)
return result_index[0]
else:
return -1
```

Kode Sumber 4.10: Implementasi fungsi *predict* pada *Knowledge-based tools* Bag. 2

Fungsi *lookUp* digunakan untuk mengubah hasil prediksi emosi yang berasal dari fungsi *get_emotion* menjadi nomor yang telah didefinisikan. Jika kata yang dicari memiliki status ”joy”, maka kata tersebut bernilai 1 atau positif. Sedangkan bila kata tersebut memiliki status ”fear”, ”anger”, ”sadness”, ”disgust”, dan ”shame”, maka kata tersebut bernilai 0 atau negatif.

Fungsi *mapEmotions* berguna untuk memetakan hasil prediksi emosi pada tiap kelas emosi yang kita gunakan. Hal ini diperlukan, karena sumber leksikal WNAffect memiliki struktur seperti *tree* dimana label emosi memiliki beberapa jenis emosi di

dalamnya. Daftar pemetaan emosi yang digunakan ditunjukkan pada Tabel 4.2. Kode Sumber 4.11 menunjukkan pengimplementasian fungsi lookUp dan mapEmotions.

```
def lookUp(emotion, mymap):
if emotion in mymap[”joy”]:
return 1
elif emotion in mymap[”fear”] or emotion in \
mymap[”anger”] or emotion in mymap[”sadness”] or \
emotion in mymap[”disgust”] or emotion in \
mymap[”shame”]:
return 0
else:
return −1
def mapEmotions():
emotions = {}
with open(’notes/joy_mapping.txt’, ’r’) as file:
mapping = file.read()
mapping = mapping.split(’,’)
emotions[”joy”] = mapping
with open(’notes/fear_mapping.txt’, ’r’) as file:
mapping = file.read()
mapping = mapping.split(’,’)
emotions[”fear”] = mapping
with open(’notes/anger_mapping.txt’, ’r’) as file:
mapping = file.read()
mapping = mapping.split(’,’)
emotions[”anger”] = mapping
with open(’notes/sadness_mapping.txt’, ’r’) as file:
mapping = file.read()
mapping = mapping.split(’,’)
emotions[”sadness”] = mapping
with open(’notes/disgust_mapping.txt’, ’r’) as file:
mapping = file.read()
mapping = mapping.split(’,’)
emotions[”disgust”] = mapping
with open(’notes/shame_mapping.txt’, ’r’) as file:
mapping = file.read()
mapping = mapping.split(’,’)
emotions[”shame”] = mapping
return emotions
```

Kode Sumber 4.11: Implementasi fungsi *lookUp* dan *mapEmotions*

4.3.3 *Training* dan *Testing* skenario data 1

Pada tahap ini, data yang kita gunakan adalah data dari fungsi *processWords*, yaitu data yang telah berisi fitur *bag-of-words* dan kelas emosi yang benar. Pada skenario data 1 ini, digunakan proses *k-cross validation* dengan nilai *k* sebanyak 10 dan jumlah data *testing* tiap iterasi sebanyak 760. Data yang digunakan pada skenario ini adalah dataset *ISEAR*. Pertama inisiasi kelas *TreeTagger* dengan parameter *TAGLANG* bernilai ”en” yang berarti menggunakan bahasa inggris. Kemudian inisiasi kelas *WNAffect* dengan memuat folder yang berisi pustaka *wordnet* dan *wordnetAffect* sebagai parameternya. Untuk Skenario pengujian 1, inisiasi kelas *MultinomialNB* menggunakan parameter *alpha* senilai 0.01 untuk mengatur *learning rate*-nya. Sedangkan untuk *Logistic Regression*, inisiasi kelas *LogisticRegression* yang berasal dari pustaka *linear_model* menggunakan parameter *solver*(*kernel*) ”liblinear”, *n_jobs* senilai 3 untuk menggunakan 3 *core* pada tahap *training*. Kemudian untuk kedua model ini, lakukan fungsi *fit* dengan parameter fitur data *training* dan kelas yang benar dari data tersebut.

Untuk tahap *testing*, buat sebuah *array* kosong dengan 2 buah elemen untuk tiap baris data dengan nilai inisial 0. Kemudian gunakan fungsi *predict* dengan model *Naive Bayes* dan *Logistic Regression*yang telah di *training* sebelumnya pada fitur data *testing* tersebut. Selain menggunakan dua model tersebut, gunakan fungsi *predict* yang telah dibuat dengan *Knowledge-based tools*. Kemudian untuk tiap hasil prediksi, tambahkan nilai sebanyak satu pada *array* kosong yang telah didefinisikan dengan indeks elemen sesuai hasil prediksi untuk menambahkan nilai *voting* pada *ensemble*. Kemudian cari nilai *voting* terbanyak pada *array ensemble* tersebut, bila nilainya seri, gunakan nilai prediksi dari *Naive Bayes*. Kemudian cek nilai prediksi dengan nilai kelas sesungguhnya. Lalu tambahkan nilai hasil prediksi pada *array* hasil prediksi. Kemudian lakukan

fungsi *prf* yaitu alias dari fungsi *precision_recall_fscore_support* dengan parameter nilai kelas sesungguhnya dari data *testing* dan nilai kelas hasil prediksi, dan *average* yang bernilai ”binary”. Kemudian catat hasilnya pada file *txt*. Kode Sumber 4.12 dan 4.13 menunjukkan pengimplementasian tahap *Training* dan *Testing* dengan skenario data 1 yang menggunakan model *Multinomial Naive Bayes* dan *Logistic Regression* dengan *kernel liblinear*.

```
bow_vector, words_num = processWords('isear')
start = 0
step = 760
end = start + step
tt = TreeTagger(TAGLANG='en')
wna = WNAffect(prefix+'wordnet1.6/',
\prefix+'wordnetAffect/')
for i in range(0, 10):
if i == 0:
data_train = bow_vector[step:, :]
data_test = bow_vector[:end, :]
elif i == 9:
data_train = bow_vector[:start, :]
data_test = bow_vector[start:, :]
else:
temp = bow_vector[:start, :]
data_train = bow_vector[end:, :]
data_train = np.concatenate((data_train, temp), \
axis=0)
data_test = bow_vector[start:end, :]
start += step
end += step
bnb = MultinomialNB(alpha=0.01).fit(data_train[:, 2:], \
data_train[:, 0])
lr = linear_model.LogisticRegression(\
solver='liblinear', n_jobs=3, max_iter=300).fit(\
data_train[:, 2:], data_train[:, 0])
total_test = 0
true_number = 0
prediction = []
```

Kode Sumber 4.12: Implementasi Tahap *Training* dan *Testing* pada Skenario 1 Bag. 1

```
for item in data_test:
score_table = np.zeros([2, 3], dtype=int)
if str(item[1]) != '0':
print item[1]
total_test += 1
result_nb = bnb.predict([item[2:]])
result_lr = lr.predict([item[2:]])
result_kb = kb.predict(item[1], tt, wna, cursor,\
'isear')
score_table[1][result_nb] += 1
score_table[1][result_lr] += 1
if result_kb != 0:
score_table[1][result_kb] += 1
predicted = np.unravel_index(np.argmax(score_table, \
axis=None), score_table.shape)
final_prediction = predicted[1]
if score_table[1][predicted[1]] == 1:
final_prediction = result_nb
if final_prediction == item[0]:
true_number += 1
prediction.append(int(final_prediction))

prediction = np.array(prediction)
if i == 9:
all_score = prf(data_test[:770,0], prediction, \
average='binary')
else:
all_score = prf(data_test[:,0], prediction, \
average='binary')
with open('ensemble_result[isear].txt', 'a') as file:
file.write("\n True :: "+str(true_number))
file.write(" from :: "+str(total_test))
file.write("\nAccuracy :: "+str((float(true_number) \
/ float(total_test))))
file.write("\nPrecision :: "+str(all_score[0]))
file.write("\nRecall :: "+str(all_score[1]))
file.write("\nF−Score :: "+str(all_score[2]))
```

Kode Sumber 4.13: Implementasi Tahap *Training* dan *Testing* pada Skenario 1 Bag. 2

4.3.4 *Training* dan *Testing* skenario data 2

Pada tahap ini proses klasifikasi sama sekali tidak berbeda dengan skenario data 1. Perbedaannya hanya ada pada data yang

digunakan, yaitu pada skenario 2 ini menggunakan dataset *AffectiveText*. Maka, pada fungsi *ProcessWords* gunakan parameter ”affective” untuk mendapatkan dataset *AffectiveText* dari database. Kode Sumber 4.14 dan 4.15 menunjukkan pengimplementasian ini.

```
bow_affective, words_num = processWords('affective')
start = 0
step = 123
end = start + step
tt = TreeTagger(TAGLANG='en')
wna = WNAffect(prefix+'wordnet1.6/', prefix+ \
'wordnetAffect/')
for i in range(0, 10):
if i == 0:
data_train = bow_affective[step:, :]
data_test = bow_affective[:end, :]
elif i == 9:
data_train = bow_affective[:start, :]
data_test = bow_affective[start:, :]
else:
temp = bow_affective[:start, :]
data_train = bow_affective[end:, :]
data_train = np.concatenate((data_train, temp),\
axis=0)
data_test = bow_affective[start:end, :]
start += step
end += step
bnb = MultinomialNB(alpha=0.01).fit(data_train[:, 2:],\
data_train[:, 0])
lr = linear_model.LogisticRegression(solver= \
'liblinear', n_jobs=3).fit(data_train[:, 2:],\
data_train[:, 0])
total_test = 0
true_number = 0
prediction = []
for item in data_test:
score_table = np.zeros([2, 3], dtype=int)
if str(item[1]) != '0':
print item[1]
total_test += 1
# ENSEMBLE
```

Kode Sumber 4.14: Implementasi Tahap *Training* dan *Testing* pada Skenario 2 Bag. 1

```
result_nb = bnb.predict([item[2:]])
result_lr = lr.predict([item[2:]])
result_kb = kb.predict(item[1], tt, wna, cursor,\
'affective')
score_table[1][result_nb] += 1
score_table[1][result_lr] += 1
if result_kb != 0:
score_table[1][result_kb] += 1
predicted = np.unravel_index(np.argmax(score_table,\
axis=None), score_table.shape)
final_prediction = predicted[1]
if score_table[1][predicted[1]] == 1:
final_prediction = result_lr
if final_prediction == item[0]:
true_number += 1
prediction.append(int(final_prediction))
prediction = np.array(prediction)
if i == 9:

all_score = prf(data_test[:116,0], prediction,\
average='binary')
else:
all_score = prf(data_test[:,0], prediction,\
average='binary')
with open('ensemble_result[affective].txt', 'a') as \
file:
file.write("\n\n\nTrue :: "+str(true_number))
file.write(" from :: "+str(total_test))
file.write("\nAccuracy :: "+str(\
(float(true_number) /
float(total_test))))
file.write("\nPrecision :: "+str(all_score[0]))
file.write("\nF−Score :: "+str(all_score[2]))
```

Kode Sumber 4.15: Implementasi Tahap *Training* dan *Testing* pada Skenario 2 Bag. 2

4.3.5 *Training* dan *Testing* skenario data 3

Pada tahap ini proses klasifikasi sama sekali tidak berbeda dengan skenario data 1 dan 2. Perbedaannya hanya ada pada data yang digunakan, yaitu pada skenario 3 ini menggunakan dataset *AffectiveText* dan *ISEAR*. Maka, pada fungsi *ProcessWords* gunakan parameter "mixed" untuk mendapatkan kedua dataset

tersebut dari database. Kode Sumber 4.16 dan 4.17 menunjukkan pengimplementasian tahap ini.

```
bow_mixed, words_num = processWords('mixed')
start = 0
step = 840
end = start + step
tt = TreeTagger(TAGLANG='en')
wna = WNAffect(prefix+'wordnet1.6/',prefix+'wordnetAffect/')
for i in range(0, 10):
if i == 0:
data_train = bow_mixed[step:, :]
data_test = bow_mixed[:end, :]
elif i == 9:
data_train = bow_mixed[:start, :]
data_test = bow_mixed[start:, :]
else:
temp = bow_mixed[:start, :]
data_train = bow_mixed[end:, :]
data_train = np.concatenate((data_train, temp),\
axis=0)
data_test = bow_mixed[start:end, :]
start += step
end += step
bnb = MultinomialNB(alpha=0.01).fit(data_train[:, 2:],\
data_train[:, 0])
lr = linear_model.LogisticRegression(solver=\
'liblinear', n_jobs=3).fit(data_train\
[:, 2:], data_train[:, 0])
total_test = 0
true_number = 0
prediction = []
for item in data_test:
score_table = np.zeros([2, 3], dtype=int)
if str(item[1]) != '0':
print item[1]
total_test += 1
# ENSEMBLE
result_nb = bnb.predict([item[2:]])
result_lr = lr.predict([item[2:]])
result_kb = kb.predict(item[1], tt, wna, cursor,\
'mixed')
score_table[1][result_nb] += 1
score_table[1][result_lr] += 1
```

Kode Sumber 4.16: Implementasi Tahap *Training* dan *Testing* pada Skenario 3 Bag. 1

```
if result_kb != 0:
score_table[1][result_kb] += 1
predicted = np.unravel_index(np.argmax(score_table,\
axis=None), score_table.shape)
final_prediction = predicted[1]
if score_table[1][predicted[1]] == 1:
final_prediction = result_lr
if final_prediction == item[0]:
true_number += 1
prediction.append(int(final_prediction))
prediction = np.array(prediction)
if i == 9:

all_score = prf(data_test[:1169,0], prediction,\
average='binary')
else:
all_score = prf(data_test[:,0], prediction,\
average='binary')
with open('ensemble_result[mixed].txt', 'a') as\
file:
file.write("\n\n\nTrue :: "+str(true_number))
file.write(" from :: "+str(total_test))
file.write("\nAccuracy :: "+str((float(true_number)\
/ float(total_test))))
file.write("\nPrecision :: "+str(all_score[0]))
file.write("\nF−Score :: "+str(all_score[2]))
```

Kode Sumber 4.17: Implementasi Tahap *Training* dan *Testing* pada Skenario 3 Bag. 2

Tabel 4.2: Daftar Pemetaan Kelas Emosi

Kelas	**Daftar Emosi**
Joy	joy, love, closeness, peace, easiness, benevolence, happiness, amusement, exuberance, happiness, bonheur, gladness, rejoicing, elation, euphoria, exultation, triumph, exhilaration, bang, titillation, contentment, satisfaction, satisfaction-pride, fulfillment, complacency, smugness, belonging, comfortableness, togetherness, merriment, hilarity, jollity, jocundity, cheerfulness, buoyancy, carefreeness
Fear	fear, negative-fear, alarm, creeps, horror, hysteria, panic, scare, stage-fright, fear-intimidation, negative-unconcern, heartlessness, cruelty, apprehension, trepidation, negative-suspense, chill, foreboding, shadow, presage, timidity, shyness, diffidence, hesitance, unassertiveness
Anger	anger, huffiness, infuriation, umbrage, dander, indignation, dudgeon, fury, wrath, lividity, annoyance, pique, frustation, displeasure, harassment, aggravation, bad-temper, irascibility, fit
Sadness	sadness, downheartedness, guilt, helplessness, lost-sorrow hopelessness, dolefulness, misery, forlornness, weepiness, cheerlessness, joylessness, melancholy, gloom, world-weariness, heavyheartedness, sorrow, regret-sorrow, attrition, compunction, guilt, repentance, lost-sorrow, self-pity, grief, dolor, mournfulness, woe, plaintiveness, depression, demoralization, dysphoria, oppression, weight, despondency, blue-devils
Disgust	disgust, repugnance, nausea

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas uji coba dan evaluasi terhadap sistem yang telah dikembangkan untuk mengklasifikasikan emosi dari data teks menggunakan metode *ensemble*

5.1 Lingkungan Uji Coba

Lingkungan penguijan sistem pada pengerjaan tugas ini dilakukan pada lingkungan dan alat kakas sebagai berikut:
Prosesor : Prosesor: Intel® Core™ i5-4200U CPU
@ 1.60GHz (4 CPUs) 2.3GHz RAM : 8192 MB
Jenis Device : Laptop
Sistem Operasi : Linux Ubuntu Desktop 16.04 LTS

5.2 Data Uji Coba

Data yang digunakan untuk uji coba klasifikasi kecenderungan emosi adalah sebanyak 10% dari total data. Terdapat tiga skenario pembagian data yang digunakan untuk *testing*. Pertama yaitu menggunakan data dari dataset *ISEAR* saja. Kedua yaitu menggunakan data dari dataset *AffectiveText*. Ketiga yaitu menggunakan data campuran dari kedua dataset tersebut.Pembagian dalam dataset sendiri menggunakan metode *k-cross validation* dengan nilai *k* sebesar 10.

5.3 Skenario Uji Coba

Subbab ini akan menjelaskan skenario uji coba yang telah dilakukan. Terdapat tiga skenario uji coba yang telah dilakukan. Metrik yang digunakan sebagai perbandingan pada tiap hasil uji coba adalah akurasi dan presisi. Nilai akurasi didapatkan dengan membagi *TP* yaitu *True Positive* dengan *True Positive* dan *False*

Negative.

$$Akurasi = \frac{TP}{TP + FN} \tag{5.1}$$

Dimana akurasi adalah rasio perbandingan jumlah prediksi yang benar terhadap total data *testing*. Sedangkan presisi adalah nilai hasil pembagian antara jumlah prediksi yang benar dengan jumlah data yang diprediksi ada pada kelas positif dengan rumus dibawah ini. Dengan menggunakan *TP* sebagai *True Positive* dan *FP* sebagai *False Positive*.

$$Presisi = \frac{TP}{TP + FP} \tag{5.2}$$

Kemudian untuk tiap skenario akan dibandingkan nilai akurasi metode *Ensemble* dengan akurasi tiap satu algoritma klasifikasi.

5.4 Skenario Pengujian 1

Pada skenario ini dilakukan uji coba klasifikasi kelas emosi dengan menggunakan dataset *ISEAR* dengan menggunakan metode *k-cross validation*, dengan jumlah data *testing* sebanyak kurang lebih 760 baris untuk tiap iterasi.

Hasil uji coba terbaik ditunjukkan saat menggunakan kombinasi data *testing* pada iterasi ke-9. Data iterasi ke-9 memiliki nilai akurasi sebanyak 93,15% dan nilai presisi sebanyak 82,22%. Selain itu, hasil uji coba terburuk ditunjukkan saat menggunakan kombinasi data *testing* pada iterasi terakhir yaitu ke-10. Pada iterasi ke-10 didapatkan nilai akurasi sebesar 89,62% dan nilai presisi hanya sebesar 71,42%.

Hasil perbandingan akurasi metode *Ensemble* dengan *Naive Bayes* dan *Max Entropy* pada dataset *ISEAR* ditunjukkan pada Gambar 5.1. Pada diagram tersebut terlihat bahwa akurasi metode *Ensemble* dapat mengalahkan performa *Max Entropy* pada enam iterasi. Namun, akurasi metode *Ensemble* dapat mencapai akurasi tertinggi dibandingkan kedua metode lainnya.

5.5 Skenario Pengujian 2

Pada skenario ini dilakukan uji coba klasifikasi kelas emosi dengan menggunakan dataset *AffectiveText* dengan menggunakan metode *k-cross validation*, dengan nilai *k* sebesar 10. Jumlah data *testing* yang digunakan pada tiap iterasi sebanyak kurang lebih 123 baris data.

Hasil uji coba terbaik ditunjukkan saat menggunakan kombinasi data *testing* pada iterasi ke-4. Data iterasi ke-4 memiliki nilai akurasi sebanyak 74,79% dan nilai presisi sebanyak 64,44%. Selain itu, hasil uji coba terburuk ditunjukkan saat menggunakan kombinasi data *testing* pada iterasi terakhir yaitu ke-6. Pada iterasi ke-6 didapatkan nilai akurasi sebesar 65,85% dan nilai presisi hanya sebesar 67,34%.

Hasil perbandingan akurasi metode *Ensemble* dengan *Naive Bayes* dan *Max Entropy* pada dataset *AffectiveText* ditunjukkan pada Gambar 5.2. Pada Diagram tersebut terlihat bahwa akurasi metode *Naive Bayes* lebih buruk daripada metode *Ensemble* dan *Max Entropy*. Kemudian dari ketiga perbandingan ini, metode *Ensemble* mengalahkan akurasi metode *Max Entropy* pada hampir semua kasus.

5.6 Skenario Pengujian 3

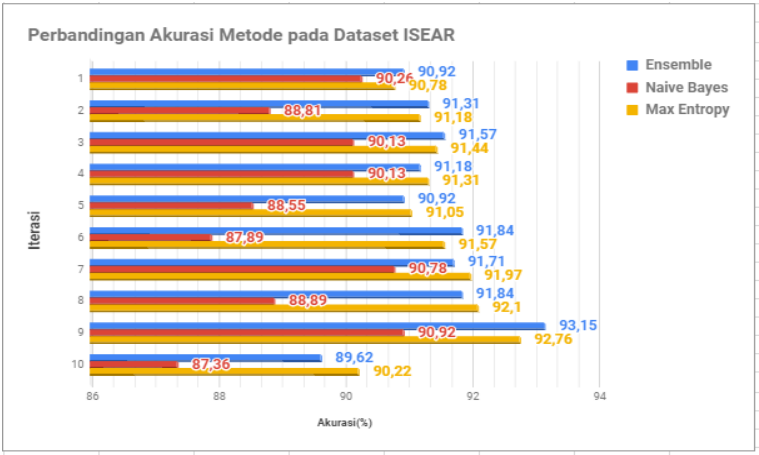
Pada skenario ini dilakukan uji coba klasifikasi kelas emosi dengan menggunakan dataset campuran antara *ISEAR* dan *AffectiveText* dengan menggunakan metode *k-cross validation*, dengan jumlah data *testing* sebanyak kurang lebih 840 baris untuk tiap iterasi.

Hasil uji coba terbaik ditunjukkan saat menggunakan kombinasi data *testing* pada iterasi ke-9. Data iterasi ke-9 memiliki nilai akurasi sebanyak 93,45% dan nilai presisi sebanyak 80,90%. Selain itu, hasil uji coba terburuk ditunjukkan saat menggunakan kombinasi data *testing* pada iterasi pertama. Pada iterasi pertama didapatkan nilai akurasi sebesar 66,19% dan nilai presisi hanya sebesar 80,45%.

Hasil perbandingan akurasi metode *Ensemble* dengan *Naive Bayes* dan *Max Entropy* pada dataset campuran *ISEAR* dan *AffectiveText* ditunjukkan pada Gambar 5.3. Pada Diagram tersebut terlihat bahwa akurasi metode *Naive Bayes* jauh lebih buruk daripada metode *Ensemble*. Kemudian dari ketiga perbandingan ini, metode *Ensemble* mengalahkan kedua metode lainnya pada hampir semua kasus dengan perbedaan yang sangat signifikan. Hal ini membuktikan metode *Ensemble* dapat beradaptasi dengan sangat baik terhadap dataset yang memiliki karakteristik data beragam didalamnya.

Tabel 5.1: Hasil Uji Coba Data Skenario Pengujian 1

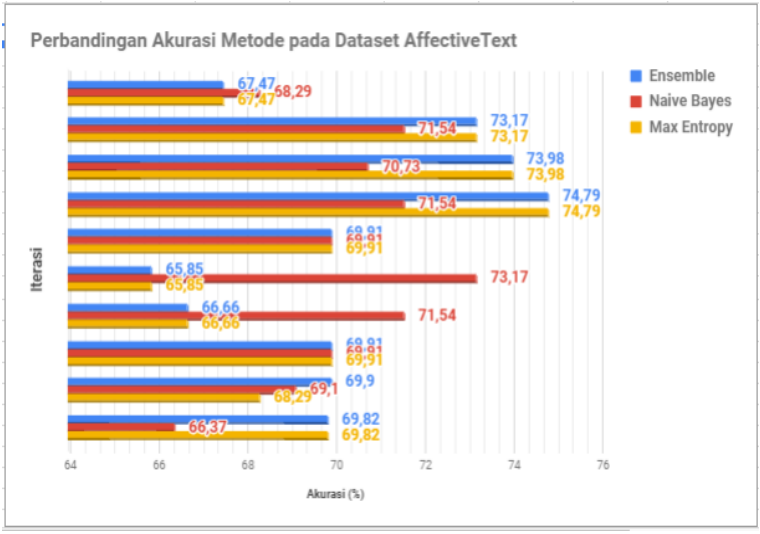
Iterasi	**Akurasi (%)**	**Presisi (%)**
1	90,92	81,94
2	91,31	82,53
3	91,57	81,42
4	91,18	76,82
5	90,92	75,67
6	91,84	90,00
7	91,71	75,58
8	91,84	80,48
9	**93,15**	**82,22**
10	89,62	71,42



Gambar 5.1: Grafik Perbandingan akurasi tiap metode pada Skenario Pengujian 1

Tabel 5.2: Hasil Uji Coba Data Skenario Pengujian 2

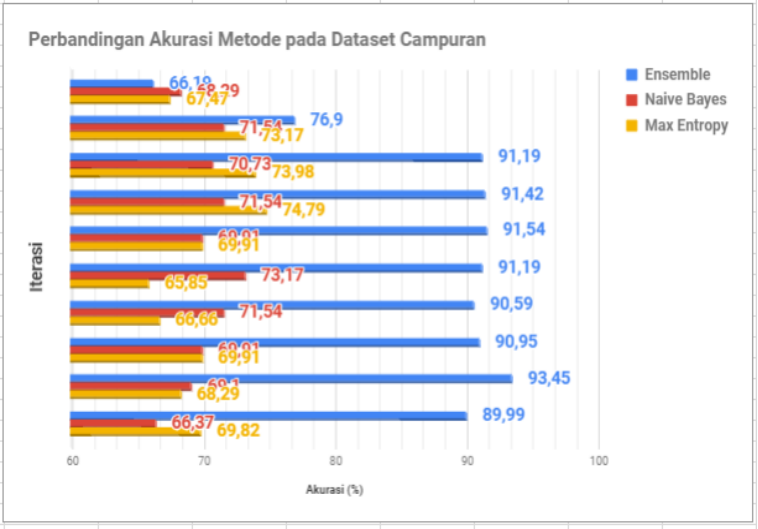
Iterasi	Akurasi (%)	Presisi (%)
1	67,47	67,44
2	73,17	64,7
3	73,98	69,56
4	**74,79**	**64,44**
5	69,91	59,25
6	65,85	67,34
7	66,66	64,28
8	69,91	88,37
9	69,9	71,11
10	69,82	69,56



Gambar 5.2: Grafik Perbandingan akurasi tiap metode pada Skenario Pengujian 2

Tabel 5.3: Hasil Uji Coba Data Skenario Pengujian 3

Iterasi	Akurasi (%)	Presisi (%)
1	66,19	80,45
2	76,9	78,07
3	91,19	81,25
4	91,42	76,74
5	91,54	78,26
6	91,19	74,44
7	90,59	75
8	90,95	69,64
9	**93,45**	**80,9**
10	89,99	68,84



Gambar 5.3: Grafik Perbandingan akurasi tiap metode pada Skenario Pengujian 3

(Halaman ini sengaja dikosongkan)

BAB VI
PENUTUP

Bab ini membahas kesimpulan yang dapat diambil dari tujuan pembuatan sistem dan hubungannya dengan hasil uji coba dan evaluasi yang telah dilakukan. Selain itu, terdapat beberapa saran yang bisa dijadikan acuan untuk melakukan pengembangan dan penelitian lebih lanjut.

6.1 Kesimpulan

Dari proses perencangan, implementasi dan pengujian terhadap sistem, dapat diambil beberapa kesimpulan berikut:

1. Pada dataset *ISEAR*, hasil uji coba terbaik didapatkan pada iterasi ke-9 dengan menggunakan metode *Ensemble* yang telah diusulkan. Metode *Ensemble* terbukti mengalahkan akurasi klasifikasi pada enam kasus dalam dataset. Metode ini mendapatkan nilai akurasi sebesar 93,15%.
2. Pada dataset *AffectiveText*, hasil uji coba terbaik didapatkan pada iterasi ke-4 dengan menggunakan metode *Max Entropy* dan *Ensemble*. Kedua metode ini mendapatkan akurasi sebesar 74,79% pada kombinasi data pada iterasi ke-4.
3. Pada dataset campuran antara *AffectiveText* dan *ISEAR*, hasil uji coba terbaik didapatkan dengan menggunakan metode *Ensemble* pada iterasi ke-9. Metode *Ensemble* mendapatkan nilai akurasi sebesar 93,45%.
4. Metode *Ensemble* dapat beradaptasi dengan sangat baik dalam mengklasifikasikan dua dataset dengan karakteristik yang berbeda.
5. Metode *Ensemble* terbukti dapat mengalahkan kemampuan klasifikasi dua model yang lain pada dataset yang sangat beragam.

6.2 Saran

Berikut beberapa saran yang diberikan untuk pengembangan lebih lanjut:

1. Pengembangan proses klasifikasi pada model *Ensemble* untuk kelas emosi yang lebih spesifik.
2. Melakukan eksplorasi pada penggunaan *knowledge-based tools* untuk dapat menganalisis struktur kalimat dan konten emosi dengan lebih akurat.
3. Menggunakan fitur *bigram, trigram,* dan *n-gram* untuk dapat menambah akurasi klasifikasi.
4. Menyeimbangkan jumlah data tiap kelas untuk meningkatkan kinerja metode klasifikasi.

DAFTAR PUSTAKA

[1] A. Kanavos, I. Perikos, P. Vikatos, I. Hatzilygeroudis, C. Makris, dan A. Tsakalidis, "Conversation Emotional Modeling in Social Networks," in *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*, Nov. 2014, hal. 478–484.

[2] V. Anusha dan B. Sandhya, "A Learning Based Emotion Classifier with Semantic Text Processing," in *Advances in Intelligent Informatics*, ser. Advances in Intelligent Systems and Computing. Springer, Cham, 2015, hal. 371–382.

[3] K. Scherer dan H. G. Wallbott, ""Evidence for universality and cultural variation of differential emotion response patterning": Correction," *Journal of Personality and Social Psychology - PSP*, vol. 67, hal. 55–55, Jul. 1994.

[4] M. Ptaszynski, H. Dokoshi, S. Oyama, R. Rzepka, M. Kurihara, K. Araki, dan Y. Momouchi, "Affect analysis in context of characters in narratives," *Expert Systems with Applications*, vol. 40, no. 1, hal. 168–176, Jan. 2013.

(Halaman ini sengaja dikosongkan)

LAMPIRAN A
KODE SUMBER PENDUKUNG

Kelas WNAffect

```
# −∗− coding: utf−8 −∗−
"""
Clement Michard (c) 2015
"""

import os
import sys
import nltk
from emotion import Emotion
from nltk.corpus import WordNetCorpusReader
import xml.etree.ElementTree as ET

class WNAffect:
"""WordNet−Affect ressource."""

def __init__(self, wordnet16_dir, wn_domains_dir):
"""Initializes the WordNet−Affect object."""

cwd = os.getcwd()
nltk.data.path.append(cwd)
wn16_path = "{0}/dict".format(wordnet16_dir)
self.wn16 = WordNetCorpusReader(os.path.abspath("{0}/{1}".format(cwd,\
  wn16_path)), nltk.data.find(wn16_path))
  self.flat_pos = {'NN':'NN', 'NNS':'NN', 'JJ':'JJ', 'JJR':'JJ', 'JJS':'JJ',\
    'RB':'RB', 'RBR':'RB', 'RBS':'RB', 'VB':'VB', 'VBD':'VB', 'VGB':'VB',\
      'VBN':'VB', 'VBP':'VB','VBZ':'VB'}
self.wn_pos = {'NN':self.wn16.NOUN, 'JJ':self.wn16.ADJ, 'VB':self.wn16.VERB,\
  'RB':self.wn16.ADV}
self._load_emotions(wn_domains_dir)
self.synsets = self._load_synsets(wn_domains_dir)



def _load_synsets(self, wn_domains_dir):
"""Returns a dictionary POS tag −> synset offset −> emotion (str −> int −> str)."""

tree = ET.parse("{0}/wn−affect−1.1/a−synsets.xml".format(wn_domains_dir))
root = tree.getroot()
```

Kode Sumber 1.1: Implementasi Kelas WNAffect

```
pos_map = { ”noun”: ”NN”, ”adj”: ”JJ”, ”verb”: ”VB”, ”adv”: ”RB” }
synsets = {}
for pos in [”noun”, ”adj”, ”verb”, ”adv”]:
tag = pos_map[pos]
synsets[tag] = {}
for elem in root.findall(”.//{0}−syn−list//{0}−syn”.format(pos, pos)):
offset = int(elem.get(”id”)[2:])
if not offset: continue
if elem.get(”categ”):
synsets[tag][offset] = Emotion.emotions[elem.get(”categ”)] if elem.get(”categ”)\
  in Emotion.emotions else None
elif elem.get(”noun−id”):
synsets[tag][offset] = synsets[pos_map[”noun”]][int(elem.get(”noun−id”)[2:])]

return synsets

def _load_emotions(self, wn_domains_dir):
”””Loads the hierarchy of emotions from the WordNet−Affect xml.”””

tree = ET.parse(”{0}/wn−affect−1.1/a−hierarchy.xml”.format(wn_domains_dir))
root = tree.getroot()
for elem in root.findall(”categ”):
name = elem.get(”name”)
if name == ”root”:
Emotion.emotions[”root”] = Emotion(”root”)
else:
Emotion.emotions[name] = Emotion(name, elem.get(”isa”))

def get_emotion(self, word, pos):
”””Returns the emotion of the word.
word −− the word (str)
pos −− part−of−speech (str)
”””
if pos in self.flat_pos:
pos = self.flat_pos[pos]
synsets = self.wn16.synsets(word, self.wn_pos[pos])
if synsets:
for synset in synsets:
offset = synset.offset()
if offset in self.synsets[pos]:
return self.synsets[pos][offset]
return None

def get_emotion_synset(self, offset):
”””Returns the emotion of the synset.
```

Kode Sumber 1.2: Implementasi Kelas WNAffect

```
offset −− synset offset (int)
"""

for pos in self.flat_pos.values():
if offset in self.synsets[pos]:
return self.synsets[pos][offset]
return None



if __name__ == "__main__":
wordnet16, wndomains32, word, pos = sys.argv[1:5]
wna = WNAffect(wordnet16, wndomains32)
print(wna.get_emotion(word, pos))
```

Kode Sumber 1.3: Implementasi Kelas WNAffect

Kelas Emotion

```
# −*− coding: utf−8 −*−
"""
Clement Michard (c) 2015
"""

class Emotion:
"""Defines an emotion."""

emotions = {} # name to emotion (str −> Emotion)

def __init__(self, name, parent_name=None):
"""Initializes an Emotion object.
name −− name of the emotion (str)
parent_name −− name of the parent emotion (str)
"""

self.name = name
self.parent = None
self.level = 0
self.children = []
if parent_name:
self.parent = Emotion.emotions[parent_name] if parent_name else None
self.parent.children.append(self)
```

Kode Sumber 1.4: Implementasi Kelas Emotion

```
self.level = self.parent.level + 1

def get_level(self, level):
"""Returns the parent of self at the given level.
level −− level in the hierarchy (int)
"""

em = self
while em.level > level and em.level >= 0:
em = em.parent
return em


def __str__(self):
"""Returns the emotion string formatted."""

return self.name


def nb_children(self):
"""Returns the number of children of the emotion."""

return sum(child.nb_children() for child in self.children) + 1


@staticmethod
def printTree(emotion=None, indent="", last='updown'):
"""Prints the hierarchy of emotions.
emotion −− root emotion (Emotion)
"""

if not emotion:
emotion = Emotion.emotions["root"]

size_branch = {child: child.nb_children() for child in emotion.children}
leaves = sorted(emotion.children, key=lambda emotion: emotion.nb_children())
up, down = [], []
if leaves:
while sum(size_branch[e] for e in down) < sum(size_branch[e] for e in leaves):
down.append(leaves.pop())
up = leaves

for leaf in up:
next_last = 'up' if up.index(leaf) is 0 else ''
```

Kode Sumber 1.5: Implementasi Kelas Emotion

```
next_indent = '{0}{1}{2}'.format(indent, ' ' if 'up' in last else ' │', '' ) *\
  len(emotion.name))
Emotion.printTree(leaf, indent=next_indent, last=next_last)
if last == 'up':
start_shape = ' ┌'
elif last == 'down':
start_shape = ' └'
elif last == 'updown':
start_shape = ' '
else:
start_shape = ' ├'
if up:
end_shape = '┤ '
elif down:
end_shape = '┐ '
else:
end_shape = ''
print('{0}{1}{2}{3}'.format(indent, start_shape, emotion.name, end_shape))
for leaf in down:
next_last = 'down' if down.index(leaf) is len(down) − 1 else ''
next_indent = '{0}{1}{2}'.format(indent, ' ' if 'down' in last else ' │', '' ) *\
  len(emotion.name))
Emotion.printTree(leaf, indent=next_indent, last=next_last)
```

Kode Sumber 1.6: Implementasi Kelas Emotion

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Hari Setiawan, akrab dipanggil Hari lahir pada tanggal 16 April 1996 di Surabaya, Jawa Timur. Penulis merupakan seorang mahasiswa yang sedang menempuh studi di Departemen Informatika Institut Teknologi Sepuluh Nopember. Memiliki hobi antara lain membaca buku psikologi dan menonton film. Selama menempuh pendidikan di kampus, penulis juga aktif dalam organisasi kemahasiswaan, antara lain Staff dan Staff Ahli Departemen Kesejahteraan Mahasiswa Himpunan Mahasiswa Teknik Computer-Informatika pada tahun ke-2 dan ke-3. Pernah menjadi staff National Logic Competition Schematics tahun 2015 dan 2016. Selain itu penulis pernah menjadi asisten dosen di mata kuliah Jaringan Komunikasi. Selain itu, penulis juga menjadi administrator di Lab Pemrograman. Penulis dapat dihubungi melalui email haristw16@gmail.com.